COMPUTER GRAPHICS IN CHEMICAL EDUCATION

A thesis submitted in fulfilment of the requirements for the award of the degree of

MASTER OF SCIENCE (HONOURS)

from

THE UNIVERSITY OF WOLLONGONG

by

ROSLYN JOY ATKINS B.Sc.(Hons),Dip.Ed.

Department of Chemistry
1987

745690

This is to certify that the work carried out in this thesis has not been submitted for a higher degree at any other university or institution.

R. Atkins

Roslyn Atkins

September, 1987.

SUMMARY

The UCSD Pascal programming language has been employed to develop three software packages suitable for senior high school and introductory university chemistry. All programs utilise microcomputer graphics and animation to illustrate specific chemical concepts. The packages, developed on the Apple II microcomputer, have been favourably evaluated in the chemistry classroom.

An important feature of the software is the presentation of a highly interactive environment in which the user can control the simulation of a variety of chemical reactions. Chemical concepts, often found difficult by students, are clearly displayed and explained through high quality graphics and animation. Most significantly, microcomputer animation in these packages provide an excellent tool to help students visualize chemical reactions at a molecular level.

A friendly user-interface is incorporated into the educationally valuable and versatile teaching packages, which are complemented with comprehensive tutorial worksheets. The packages are designed to be either used by a teacher for classroom demonstrations, or by students for individual tutorials.

CONTENTS

CHAPTER 1. INTRODUCTION

1.1 BRIEF OVERVIEW OF RESEARCH PROJECT

1.2 BACKGROUND - COMPUTERS AND CHEMICAL EDUCATION

1.3 FEATURES OF MICROCOMPUTER EDUCATION

- 1.3.1 Interactive instruction
- 1.3.2 Graphics and animation
- 1.3.3 Computational capabilities

1.4 APPLICATION OF MICROCOMPUTERS IN CHEMICAL EDUCATION

- 1.4.1 Classroom demonstrations
  - Graphical representation of chemical data
  - Emulation of chemical instrumentation
  - Simulation of laboratory experiments
  - Simulation of atoms and molecules
- 1.4.2 Individualized instruction
  - Drill and practice
  - Tutorial CAI
  - Simulations
- 1.4.3 Student assessment
- 1.4.4 Information storage and retrieval
- 1.4.5 Data acquisition and instrument control

1.5 REVIEW OF GRAPHICAL SIMULATION PROGRAMS

- 1.5.1 Graphical simulations of titration experiments
- 1.5.2 Graphical simulations of chemical reactions

1.6 CURRENT STATUS MICROCOMPUTERS IN CHEMICAL EDUCATION

- 1.6.1 Hardware
- 1.6.2 Software

1.7 OBJECTIVES OF RESEARCH PROJECT

1.1 BRIEF OVERVIEW OF RESEARCH PROJECT

The aim of this project is to use the graphics facilities of a microcomputer to illustrate specific chemical concepts. The topics considered are: qualitative and quantitative description of titrations between acidic and basic solutions; qualitative description of two oxidation/reduction reactions; and qualitative description of two acid/base reactions. As a result three software packages suitable for senior high school chemistry have been developed. All programs are written in UCSD Pascal for the Apple II family of computers and require a minimum 64K bytes of random access memory (RAM). The three packages are designed to be either used by a teacher for classroom demonstrations, or by students for individual tutorials. Each package has been trialled in the chemistry classroom.

Briefly, the purpose of the programs presented in this thesis are:

(i) Acid/Base Titration Package.

This package contains a series of simulations relating to the titration of acids and bases, with particular emphasis being placed on the selection of appropriate acid/base indicators.

(ii) Salt Titration Package.

This package extends upon the previous package by providing a series of titration simulations relating to acidic and basic salts and mixtures of salts. Emphasis is placed on developing problem solving skills.

(iii) Macroscopic/Microscopic Chem Demonstrations.

This package simulates four common chemical reactions, two acid/base reactions and two oxidation/reduction reactions. The simulations show the four chemical reactions at both microscopic and macroscopic levels.

A comprehensive series of tutorial worksheets has been designed to complement the programs in these three packages.

1.2 BACKGROUND - COMPUTERS AND CHEMICAL EDUCATION

Early chemical education programs were almost always developed on expensive mainframe computers and have been used for teaching chemistry for over two decades. One of the most famous computer aided instruction (CAI) projects has been the PLATO system set up at the University of Illinois (Alpert et al.(1970)). The first PLATO system was developed in the early sixties and since then a great variety of software has been developed for educational and training purposes. The PLATO system has been a required part of several chemistry courses at the University of Illinois for over a decade. (Smith (1976); Wiegers et al.(1980))

The PLATO system is a comprehensive hardware/software system designed specifically for graphics-oriented instructional applications. Special terminals are used which incorporate touch-sensitive plasma-panels to facilitate student interaction (Smith et al. (1976)). The role of PLATO in these courses is twofold: first, it provides interactive individual instruction on the basic course material; and second, it furnishes an automated course outline, a weekly assignment schedule, answers to exam questions, provides a communication medium between students and the instructor, and provides a report for each student on his grades and class standing.

Chemical educators at universities are continuing to develop educational systems on mainframes. Unfortunately, due to the high cost of graphics terminals, much of this software does not incorporate graphics.

In the late 1970's and early 1980's a wide variety of inexpensive microcomputers became available. The last ten years has seen a considerable decrease in the cost of microcomputers and a corresponding proliferation of educational software. Whereas mainframes restricted CAI largely to tertiary institutions, microcomputers, due to low cost, have made CAI accessible to all levels of education. Furthermore, students and chemistry teachers, have found microcomputers much "friendlier" to use than mainframes. Many users are intimidated by such things as logging-on, passwords and operating systems associated with mainframes (Dessy (1982)). One of the greatest advantages of the microcomputer over the mainframe is the facility to produce high quality graphics very readily (Breneman (1981)).

Microcomputers are being used in increasing number, in secondary and tertiary chemistry courses, to provide direct tutorial instruction, pre-lab and post-lab simulations, collection and processing of laboratory data, lecture demonstrations, class management and control of videotapes and video disks (Dessy (1982); Kolodny (1983); Smith (1984); Russell (1984); Gerhold (1985); Settle (1987)).

1.3 FEATURES OF MICROCOMPUTER EDUCATION

A computer can be utilised as an interactive medium for instruction unlike anything possible with a textbook. The interactive character of instructional material combined with the graphics and computational capabilities of the computer offer exciting new methods in chemical education.

1.3.1 Interactive instruction

Interactive instruction is possibly the most important single feature of computer instruction (Bork (1984)). The interactive nature of microcomputers enables the student to become intrinsically involved in the lesson presented. This active mode of learning contrasts with the passive mode encountered in most (chemistry) lectures. The computer is able to provide immediate feedback - student input can be processed instantaneously and the student informed whether the response is correct or not.

Interaction between the computer and a student can result in individualized tuition, with the computer formulating a path of instruction determined by the responses of the student. The rate of the instruction can be controlled by the student enabling progression at a pace appropriate for each student.

1.3.2 Graphics and animation

Graphics and animations may be utilized to make CAI very different to that of the static printed page in a book. Chemistry teachers have found visual aids to be extremely useful (if not essential) for explaining chemical concepts. Such concepts often involve an environment which the student, under normal circumstances, will not have the opportunity of observing

directly. For example, atoms and molecules are well suited to illustration by models, graphics and animation. The microcomputer can display these phenomena interactively using graphics or animation, thereby making it an extremely powerful visual aid in chemical education.

A system integrating the microcomputer with the videodisk offers great potential for computer aided instruction. Several interactive videodisks have been prepared for chemical instruction (Russell (1984)), however interactive video technology is presently restricted by the high cost of the hardware components and the controlling software.

1.3.3 Computational ability.

The computer can be programmed to handle a great variety of scientific calculations. One problem in teaching chemistry, particularly physical chemistry, is that the mathematical ability of many students is often inadequate to handle the complex calculations necessary to understand the topic. Furthermore, most common calculators are inadequate for such calculations . Computer instruction can help students master quantitative aspects of physical chemistry by allowing the student to manipulate and analyse data without being inhibited by the complicated mathematics involved. This is exemplified in the teaching of quantum chemistry where the student can become so preoccupied with the computations, that the overall concepts are not appreciated (Barrow (1980); Joshi (1985)).

With the advent of commercially available spreadsheets, many

chemical educators have utilized them to assist in tedious chemical calculations (Coe (1987); Luibrand et al.(1987)).

1.4 APPLICATIONS OF MICROCOMPUTERS

Common applications of microcomputers in chemical education include:

- classroom demonstrations.
- individualized instruction.
- student assessment.
- information storage and retrieval.
- data acquisition and instrument control.

These five applications are discussed with greater attention being given to the microcomputer as a tool for classroom demonstrations and individualized instruction.

1.4.1 Classroom demonstrations

The microcomputer, used with an appropriate display device can function as an electronic blackboard. Most of the current microcomputers can output a video signal to a standard TV set. Several TV sets can be driven by a single microcomputer, or if the facilities allow, a large projection monitor can be used. The major advantages of this technology over the traditional blackboard (or overhead projector) are due to the graphics and computational capabilities of the microcomputer. Computer demonstrations illustrate chemical concepts, encourage class involvement and make the lesson more interesting. The majority of classroom demonstrations are simulations, which may be classified into four broad groups:

(i) Graphical representation of chemical data.
(ii) Emulation of chemical instrumentation.
(iii) Simulation of laboratory experiments.
(iv) Simulation of atoms and molecules.

(i) Graphical representation of chemical data

The computer can simulate experiments in the sense that simulated data are tabulated and/or plotted. Graphs may be produced almost immediately and the flexibility of the data can be considered. The effect of changing various parameters can readily be illustrated (Johnson(1980)). Only the simplest graphics is required for such demonstrations.

Numerous programs of this type have been developed, including plotting acid/base titrations curves (Breneman (1981)); illustration of the relationship between spectral amplitude and frequency in nuclear magnetic spectra (Newmark (1983)); graphical representations of solutions to Schrodinger equation as a function of approximate well potentials (Kubach (1983)); analysis of enzyme kinetic calculations (Adams et al.(1984)); kinetics involved between macromolecules and smaller ligands (Dombi (1984)); calculation of overlap integrals for one-electron wave functions (Geanangel et al.(1986)); and analysis of vibrational spectra of iodine (Armanious (1986)).

(ii) Emulation of chemical instrumentation

A simulation can emulate the control and output from a chemical instrument. A realistic or schematic instrument may be displayed. The demonstration can illustrate how the instrument works; how to optimize operation of the instrument; and the relationship between instrument output and chemical nature of sample. Simulations have been used to emulate most chemical instruments including spectrophotometers (ultra-violet, visible and infra-red) (Gilbert et al.(1982)); mass spectrometers

(Brownawell et al.(1982)); nuclear magnetic resonance spectrophotometers (Draper et al.(1984); Starkey(1986)); and gas chromatographs (Whisnant (1983)).

(iii) Simulation of laboratory experiments

A simulation of a laboratory experiment (which may or may not be carried out by the students) can be used to link theoretical chemical concepts with practical considerations. For example, in an organic chemistry lecture, the teacher can illustrate the practical methods involved in the synthesis of a compound (such as solvent extraction, recrystallisation, and distillation), in addition to specifying the appropriate chemical equations. High quality graphics and efficient computational abilities are required to produce good laboratory simulations.

(iv) Simulation of atoms and molecules

A simulation displaying atoms and molecules colliding in a chemical reaction illustrates chemical processes more effectively than any other method. The option of freezing the animation to closely examine a particular aspect of the reaction makes computer simulations a most powerful tool in chemical education. Very few simulations of this type have been developed, mainly due to the large amount of programming effort required to simulate the animation of complex objects (Bendall (1987)).

Several molecular simulations illustrate the stereochemistry of molecules, by allowing rotation of a three dimensional representation of the molecule (Nakano et al.(1983); Hull (1983); Howber (1985); Farrell (1987)). Simulations have been used to

illustrate chemical bonding (Pankuch (1984)), polymer configuration (Bishop (1986)) and protein structure (Rhodes 1986)).

1.4.2 Individualized instruction

Most CAI packages are designed to be used by students working individually, however, some packages are flexible enough to be suitable for either small group or entire class use. Individualized CAI may be broadly classified intro three modes: drill and practice; tutorial; or simulation. Frequently more than one mode of instruction is employed in a CAI program. The following commercial CAI packages use all three modes of instruction:

Introduction to Organic Chemistry: Smith (1981);
Concentrated Chemical Concepts: Cornelius (1983);
Computer Aided Instruction:General Chemistry: Butler (1983);
Pre-lab Studies for General Organic and Biological Chemistry: Olmsted (1984);
Introduction to General Chemistry: Smith et al.(1985);

(i) Drill and practice.

Drill and practice software is a most popular mode of computer instruction, largely because it is the easiest to develop. Generally, the student has previously been exposed to the content of the lesson (by lecture or textbook) and the computer is used to reinforce the material. Drill and practice is highly interactive, with immediate feedback on the correctness of an answer. Drill and practice is well suited to a number of areas of chemistry in which the student must remember a number of facts or rules. Programs have been developed for a wide variety of chemistry

topics. *Symbols to Moles* (Brandwood et al.(1982)) is a comprehensive commercial package consisting of fourteen drill and practice programs on topics including chemical symbols, valencies, formulae, equations and stoichiometry. *The Chemistry Tutor*[1] (Rinehart(1985)) is an excellent commercial drill and practice package for stoichiometric calculations.

Classroom drill is very time consuming, and generally is not designed to suit all levels of student abilities. Computer drill and practice can cater to the individual ability of each student. The amount and difficulty of drill and practice problems can be determined by the student. Microcomputer drill and practice is very popular with chemistry students and performs an essential role in chemical education.

Chemistry computer games are usually a variation of drill and practice, with the program catering for several users. A number of chemistry games have been developed for topics such as chemical elements (Fleisher (1984)); chemical symbols (Ryan (1986)) and organic synthesis (Flash (1985)).

(ii) Tutorial CAI

In tutorial CAI the computer functions as the instructor, sometimes presenting new material to the student and sometimes reviewing material. Typically, tutorial CAI follows programmed instruction methodology. A small amount of information is presented in a frame (or series of frames) to which the student must respond. The flow of the lesson is determined by student

1: CHEMISTRY TUTOR has been rated one of the best educational software packages by a panel of reviewers.(Klopfer(1986)).

responses. An incorrect response may result in prompting the student to try again; providing the correct response; displaying some earlier frames or providing remedial instruction. In this way, tutorial CAI provides individualized instruction.

The major problems with tutorial CAI are that student responses are frequently limited to letters or numbers, and much of the software provides inadequate analysis of student responses. Tutorial CAI has been developed for most chemical concepts including organic synthesis (Bertrand et al.(1986); Flash (1987)). *Introduction to General Chemistry* is a high quality commercial tutorial package developed by Stanley Smith, Ruth Chabay and Elizabeth Kean. The ten disks in the package contain programs on periodic table, names of elements, istopes, inorganic nomenclature, balancing chemical equations, molecular weights, empirical formulae, ideal gases, acid-base reactions and solubility.

(iii) Simulations

All types of simulations used in classroom demonstrations (Section 1.4.1) can be incorporated into CAI suitable for individual or small group instruction. The most important feature of interactive computer simulations is it allows the student to investigate and test hypotheses. Computer simulation of laboratory equipment and experiments is particularly appropriate in situations where the equipment is not available or where the experimental method is too hazardous or difficult.

Microcomputer simulations are ideal for prelaboratory

instruction. Such simulations allow the student to focus on the critical elements of the experiment which are highlighted in the simulation. Pre-laboratory simulations ensure that the students are well prepared for the laboratory session. Post-laboratory or laboratory-extension simulation enable the student to reinforce the concepts and techniques acquired in the laboratory.
Further, computer simulations can place the student in the role of an industrial or environmental chemist and enable the manipulation of a great range of chemical instruments. This type of simulation is excellent as it enables the integration of various chemical techniques such as sample collection, wet and instrumental analysis, and searching chemical data.

Simulations for individual use have been developed for a variety of chemistry topics including plotting experimental results (Suelter et al.(1981); Brown (1982); Simpson (1986); Tirri et al.(1986)); chemical instrumentation (Suelter et al.(1981); Cabrol (1985)); pre-laboratory instruction (Olmsted et al.(1983)); simulating hypothetical situations (Whisnant (1984); Bauder (1985)); acid-base titrations (Frazin et al.(1983); Holdsworth (1986)); equilibrium (Simpson (1986)); and Boyle's Law (Suder (1983)). *Chem Lab*, published by Simon & Schuster (1985) is a high quality commercial package which uses graphics to simulate a wide variety of high school level laboratory experiments.

1.4.3 Student assessment

Microcomputers can provide student quizzes and diagnostic tests with the option of recording the students results on disk (Waught (1987)). Adequate student preparation for laboratory classes can be encouraged by requiring students to undertake a computer quiz before commencing practical work (Kolody et al.(1983)). Well designed computer quizzes are fun and motivating for students and far less threatening than the traditional test environment.

With a large number of students, it may not be possible to provide sufficient computer access time for each student, but the microcomputer can still be used to mark quizzes and record results. A card reader interfaced to a microcomputer enables rapid evaluation of student quizzes (Bath et al.(1983)). Quizzes are restricted to multiple choice or problems with a numerical answer. This is obviously advantageous in university situations where large numbers of chemistry students are involved.

1.4.4 Information storage and retrieval

Chemical data can be stored on a computer disk or tape with large amounts of related information stored in the form of a database (Rusch (1981). The microcomputer can be used as a means of rapidly analysing and retrieving the chemical information. The student can specify criteria of the desired information and the computer will analyse all information in the database, displaying only information which satisfies this criteria. Computer manipulations on large amounts of chemical data allow the student to observe trends or similarities which would be tedious to observe by other means (Wood (1986)).

would be tedious to observe by other means (Wood (1986)). Students have used commercial database packages to produce a mass spectral library (Gouge (1984)) and chemical databases have been incorporated into CAI material (Whisnant (1984); Feng et al.(1986); Biro et al.(1986)).

Microcomputers can be used as terminals to mini- or mainframe computers providing access to larger databases.

1.4.5 Data acquisition and instrument control

Microcomputers can be used as tools for data acquisition and analysis and the control of instruments. Many microcomputers can be interfaced to scientific instruments either through a serial port or through specialised interface modules such as analog to digital converters. There are obvious advantages to automatic data collection by microcomputers in experiments which require rapid or time sensitive sampling. Data collected by the microcomputer may be stored in a form which can readily be manipulated and displayed. The microcomputer can provide rapid graphical displays of raw and processed data.

Microcomputers have been interfaced to a variety of instruments, for the purpose of educational experiments, including an automated pH-stat and titrator for kinetics reactions (Cornelius et al.(1983)); a photodetector for flash photolysis experiments (Traeger (1981)); and a potentiometer for kinetics experiments (Horst et al.(1986)). Moore (1986) discusses several experiments which interface the microcomputer to thermistors and LEDs. An excellent commercial software/hardware package, *Science*

Toolkit(Shumway et al.(1985)) is specifically designed for high school science experiments.

1.5 REVIEW OF GRAPHICAL SIMULATIONS PROGRAMS

Possibly the best use of microcomputers in chemical education lies in the combination of animation and interaction in the form of simulations. This project has involved the development of several simulation programs using high quality graphics. These programs fall into two categories:

(i) simulation of titration experiments.

(ii) simulation of chemical reactions at macroscopic and microscopic levels.

Several programs similar to those developed in this project are reviewed.

1.5.1 Graphical simulations of titration experiments

A number of public domain titration simulations have been developed (Gelder (1980); Breneman (1981); Holdsworth (1986)), however commercial programs are nearly always of much higher quality. A review of two commercial titrations programs is given: *Titration Simulation* and *Titration Experiment*

Titrations Simulation [2] by J. Frazin & Partners (1983)

The first part of the simulation involves several frames of instructions. The aim of the titration is to determine the concentration of sodium hydroxide by titration with standardized hydrochloric acid. A 40mL volume of unknown is placed into a beaker. The user may select the concentration of hydrochloric acid (from a choice of four concentrations) or allow the program to determine the concentration.

2: *Titration Simulation* is one of three programs belonging to *Chemistry: Acids and Bases* package published by Encyclopedia Britannica (1983).

The simulation involves two graphics displays, with the user being able to toggle between the two. The first display illustrates a beaker and the bottom half of a burette, and the second display illustrates a close up section of the burette showing the meniscus. The close up view is required so that the user may read the burette scale and determine volume of titrant used.

The simulation proceeds by pressing the space bar, which releases 0.05mL titrant. The titration is speeded up by holding the repeat key. An indicator changes colour at the end point. Based on the volume of acid required to reach end point, the user must calculate, and enter, the concentration of the sodium hydroxide solution. The user is informed of the "percent error" in this value and is provided with the option of repeating the titration; seeing the correct answer; or trying a new titration.

The package should be considered as a poor example of CAI:

(i) The graphics is slow and of poor quality. The simulation was viewed on a black and white monitor and it was difficult to read the scale of the burette. The burette initially appeared to be empty as it was shaded the same as the background. The program continually redrew the graphics screen every time the user toggled between the two graphics displays.

(ii) The user is not given the option to experiment within the simulation. The only parameter that can be altered is the concentration of the standard solution, which can only be selected from four provided values. The unknown is always

40mL of sodium hydroxide which is always placed in the beaker.

(iii) This program provides a very low level interaction.

(iv) There is no simple mechanism for exiting from the simulation.

(v) The instructions made no mention of the indicator which is illustrated during the simulation.

(vi) Viewing the instructions cannot be avoided on subsequent use of the program.

This program has also been reviewed by Victor Bendall and Robert Roe in the Journal of Chemical Education (August,1986). Bendall gave the program, and the entire package, an extremely poor review. Roe appeared to approve this program, but made several comments for improving the package as a whole.

Titration Experiment[3] by Stanley Smith and Elizabeth Kean (1984)

This program consists of three parts:

1. Introduction, which presents a short tutorial on equivalent weights.
2. Procedure, which briefly outlines the general procedures for doing an acid base titration for determining an equivalent weight.
3. Titration Experiment. The student must conduct a simulated titration to determine the equivalent weight of an acid. Briefly, the following steps are involved in the titration:
   (i) The burette is filled with 0.100M sodium hydroxide, and the user must adjust the level to an appropriate value and then enter the inital burette reading.
   (ii) The user must specify the mass of unknown acid, within specified range, which is to be dissolved in water and placed in the beaker.
   (iii) The user is then prompted to add phenolphthalein.
   (iv) The program automatically initiates the titration, the speed of which is adjusted by pressing 'F' (faster) or 'S' (slower) until end-point is reached. The user must enter the final burette reading.
   (v) Calculations involved in determining the equivalent weight with the values obtained are presented, from which the user may identify the acid from a list of known equivalent weights.

3: *Titration Experiment* is one of many programs belonging to *Introduction to General Chemistry* package published by COMPress. *Titration Experiment* is found on Disk 8 which is titled *pH:Acids and Bases in Water.*

This is a tutorial type simulation in which the program closely controls the responses of the user. For example, if the titration is proceeding rapidly near the end-point, the program reminds the user to slow down. This program, indeed the entire package, uses high quality graphics and involves a high level of student interaction. The package is menu driven and the user interface is friendly, providing an escape facility from any point in the program. This program should not be viewed in isolation from the other programs on the same disk - the eight programs are integrated into an extensive tutorial on acidic and basic reactions.

Although *Titration Experiment* is undoubtedly of much higher quality than *Titrations Simulation*, both simulations are severely limited in that they permit the user only token control over the titration. The user is able to select the concentration (or mass) of an acid and the program determines all other titration variables. Both programs use the simulation primarily to determine the acidity of a particular sample, without enabling the user to investigate the numerous variables involved in a titration.

Furthermore, these programs focus on titration techniques which are probably better mastered in the laboratory, such as reading a burette, whilst neglecting features which are well suited to computer simulation. Obviously, calculation of pH values is an ideal task for computer determination, yet this receives little attention. Many other features, such as the suitability of an acid/base indicator could be readily illustrated within the simulation.

1.5.2 Graphical simulations of chemical reactions

Several programs use graphics to illustrate molecules (Section 1.4.1(iv)). However very few programs attempt to animate molecules to simulate chemical reactions. Stanley Smith has incorporated animation of chemical reactions into a number of commercial tutorial CAI programs (Smith(1981); Smith et al.(1985)). Typically Smith uses animation as a minor feature of the programs, however one program, *Reactions of Alkenes*, uses animation to a greater extent. *Reactions of Alkenes* is reviewed as an example of animation employed to simulate chemical reactions.

Reactions of Alkenes[4] by Stanley Smith (1981)

This program contains three sections, each illustrating a typical alkene addition reaction:

Hydrogenation.

Firstly the equation for the conversion of propene to propane is given, however the following animation refers to the conversion of ethene to ethane. Preferably, the equation and simulation should refer to the same reaction. The reaction between ethene and hydrogen using a platinum catalyst is animated as follows:

(a) hydrogen molecules strike the surface of platinum, and are adsorbed onto the surface as hydrogen atoms

(b) an ethene molecule approaches the platinum surface whereby two hydrogen atoms add on forming an ethane molecule.

The hydrogen molecules are represented as "H-H"; the platinum

4: *Reactions of Alkenes* is one of many programs belonging to *Introduction to Organic Chemistry* package published by COMPress. *Reactions of Alkenes* is found on Disk 1 which is titled *Alkanes & Alkenes*

surface is represented as a row of "..Pt-Pt-Pt..."; and ethane and ethene are represented by the appropriate structural formulae.

Halogenation.

After presenting the equation for the bromination of ethene to form 1,2-dibromoethane, the following reaction is animated:

(a) Ethene is displayed, representing the pi bond as two overlapping p orbitals.

(b) A bromine molecule strikes the ethene molecule causing Br^+ to add on, forming a bromonium ion, with Br^- breaking away.

(c) A bromide ion approaches the bromonium ion from the opposite side of the pi-bond, adding on to form 1,2-dibromoethane.

This animation is followed by a tutorial segment which graphically illustrates (without animation) that a chloride or hydroxide ion could have added on in step (c) forming the chloro -bromo product or bromo-alcohol product respectively.

Hydrogen halide addition

The animation displays a hydrogen ion adding onto propene to form a secondary carbonium ion. The rest of the reaction (addition of chloride ion) is displayed graphically, but without animation. Graphic equations show that addition of a bromide ion to the carbonium ion yields t-butyl bromide whilst addition of hydroxide yields the tertiary alcohol. Finally, a tutorial segment discusses the relative stability of primary, secondary and tertiary carbonium ions.

The graphics and animation used in this program successfully

illustrate the three reactions considered. However, the animation segments are relatively short without provision to freeze the animation during the reaction. Animation involved movement of simple shapes, such as structural formulae, with only one shape being moved at a time.

1.6 CURRENT STATUS OF MICROCOMPUTERS IN CHEMICAL EDUCATION

Utilisation of microcomputers in teaching chemistry is well below its current potential. The major problems inhibiting more widespread use are:

(i) Insufficient good quality software (Schibeci (1985); Klopfer (1986)).

(ii) Insufficient access to microcomputers by chemistry teachers (Moore (1987)).

1.6.1 Hardware

As the cost of hardware continues to fall, schools will acquire a greater number of machines. It is essential that microcomputers are available in the chemistry classroom so that teachers may utilize and experiment with computer based education. In many chemistry classrooms, a microcomputer is not always readily available. Inconvenient, or unreliable, access to computers may discourage the chemistry teacher from using a microcomputer. John Moore (1987) notes that the common practice of siting school computers in a specialized computer room is stifling the use of computers in areas such as chemistry.

1.6.2 Software

Most of the early microcomputer software was predominantly text-based computer aided instruction, following on from the systems developed on the mainframes. To date, insufficient use has been made of the graphical features of the microcomputer, one of the major reasons being the great amount of time required to develop graphics programs. Good quality graphics chemical simulations and chemical modelling programs are rare. Some programs which make good use of graphics do so to catch

attention rather than convey some chemical concept.

A major problem with educational software has been the extremely unfriendly user-interface. Many CAI programs (including commercial programs) lack several of the following features essential for a friendly interface:

(i) an escape or quit option which the user can activate at any point in the program.

(ii) range checking routines for numerical input.

(iii) indication of type of response required.

(iv) editing features for user input.

In recent years consultation between computer programmers, chemistry educators, research chemists and learning experts has resulted in an increase in quality of software (Moore et al.(1984)). Unfortunately, development of educational software on microcomputers has been hampered by the large number of incompatible microcomputers available. Furthermore, there is little incentive for the commercial development of educational software due to the typically low financial returns.

1.7 OBJECTIVES OF RESEARCH PROJECT

This research project involved the development of a series of microcomputer educational programs, suitable for high school chemistry, based on the following objectives:

1. Computer graphics and animation should be utilized to simulate chemical reactions.
2. The programs should be useful as classroom demonstrations and as tutorials for individual students.
3. A highly interactive environment should be provided.
4. Concepts which are difficult to illustrate by traditional means should be displayed.
5. An extremely friendly user-interface must be provided, with the inclusion of the following features:
   - expected responses should be made clear to the student; The type of response should be indicated (e.g. "Y/N") and the acceptable range of numerical input should be presented (e.g. "0.100 - 1.00")
   - analysis of students responses should prevent the system from crashing due to unrealistic input;
   - the student must be able to exit from the program at any time;
6. Programs should provide teachers with an interesting and motivating teaching aid.

Selection of chemistry topics

1. Acid/base titrations

The topic of acid/base titrations was selected as it involves a number of fundamental chemical concepts and also deals with a

most important method of chemical analysis. Most students carry out a small number of titrations during high school chemistry and many students commence university chemistry with only a superficial understanding of the topic.

The titration technique is well suited to microcomputer animation however, currently available programs are extremely limited in the concepts displayed. Existing titration programs allow the user to determine the value of only one (or at the most two) of the numerous variables involved in the titration. (Section 1.5.1)

The main objective of the titration programs is to allow investigation of various titration features by placing the user in control of the simulation. This can only be achieved by allowing the user to determine: the concentration and strength of both solutions; which solution is to be titrant; volume of solution to be placed in flask; and which indicator to use.

<u>2. Simulation of simple chemical reactions.</u>

Often high school students have great difficulty in visualizing chemical reactions. Consequently, students frequently memorize chemical equations rather than understand the corresponding reactions. Microcomputer animation is undoubtedly an excellent tool to help students visualize chemical reactions at a molecular level. Unfortunately very few programs have been developed for this purpose.

Four fundamental chemical reactions were selected for

microcomputer animation:

(i) reaction between an acid and an active metal;

(ii) reaction between an acid and a carbonate;

(iii) reaction between water and a very active metal;

(iv) reaction of litmus with both acidic and basic solutions.

The major objective of these programs is to display each reaction at a molecular level focusing on , and also to link the microscopic view of the reaction with the macroscopic view.

CHAPTER 2 UCSD PASCAL

2.1 INTRODUCTION TO PASCAL

Microcomputer programs are usually written in BASIC as this is the on-board language of most microcomputers. However BASIC has many features which make it unsuitable for the development of lengthy programs. In particular, large BASIC programs are extremely difficult to read and modify.

Pascal is a general purpose, high level programming language originally designed by Niklaus Wirth in the late 1960's. A more recent version of Pascal developed at the University of California at San Diego (UCSD) has gained widespread acceptance for use on microcomputers (Zaks (1981)). Pascal is generally preferred in the development of long and complex programs because of its highly structured nature and its easily read source code. Furthermore, UCSD Pascal facilitates the development of extremely large and complex programs through features such as units, segments and linking. Programs presented in this thesis are written in Apple Pascal Version 1.2.

2.2 LIBRARY UNITS

Program development of the packages prepared in this project was simplified by the use of Pascal units. A unit is a separately compiled module of code which is stored in a library file. Typically, a unit contains a group of data structures and a number of procedures and functions relating to a common application. Any program can access the routines contained within a unit by naming the unit in the USES declaration part of the application program. Utilizing library units reduces the amount of code that needs to be compiled during program development.

There are two different types of units: regular units and intrinsic units. A regular unit is incorporated into the code of an application program by being linked at the time of compilation. An intrinsic unit is loaded from the library file into memory at the time of execution. Intrinsic units must be available in a library file whenever an application program is compiled or executed.

Apple Pascal provides several specialized units in SYSTEM. LIBRARY. Two of these library units, Turtlegraphics and Chainstuff are utilized by all programs developed in this project. The Apple Pascal Language Reference Manual gives a detailed description of all routines in these units.

Three additional customized library units have been developed:

(i) Useful Unit is a general purpose unit used by all three packages developed in this project;

(ii) Titrlib Unit provides routines which are specifically required by programs in the Acid/Base Titration package;

(iii) Saltlib Unit provides routines which are specifically required by programs in the Salt Titration package.

2.2.1 TURTLEGRAPHICS UNIT

Apple Pascal provides the Turtlegraphics unit for access and manipulation of the graphics screen. The graphics screen is regarded as an array of 280 (horizontal) by 192 (vertical) pixels. "Turtle" graphics was first introduced by Seymour Papert of Massachusetts Institute of Technology in 1969 in the form of Logo programming language (Papert(1980)). A shape is drawn by moving a so-called "turtle" around the screen. The "turtle" supposedly drags a pen which leaves a trail in its path. Most of the routines provided in the Turtlegraphics units are "turtle" commands.

The Turtlegraphics unit is further discussed in Section 2.3.

2.2.2. CHAINSTUFF UNIT

The menu driven feature of the packages developed relies upon routines from the Chainstuff unit. Procedure SETCHAIN allows one program to "chain to" another program. When the first program has completed execution, a specified program will then be executed.

The chainstuff unit also enables a message to be stored in memory by one program for subsequent retrieval by other programs. This feature is used by all packages (developed in this project) to record the type of monitor in use. The user is asked once, at the start of each package, whether a colour monitor is available. Procedure SETCVAL is used to record data on the type of monitor, and as programs are selected from the package, procedure GETCVAL is implemented to retrieve the information.

2.2.3 USEFUL UNIT

The Useful unit is a general purpose unit used by all three packages. Routines in this unit are designed to perform a variety of tasks. In particular, a number of routines are related to facilitating user input.

Declaration Section

```
UNIT USEFUL;  INTRINSIC CODE 25 DATA 26;
```

The declaration indicates that USEFUL is an intrinsic unit for which the code occupies operating system segment 25 and the data occupies segment 26.

Interface Section

```
INTERFACE
USES TURTLEGRAPHICS;
CONST
   SPACE=' ';
   XMIN=0; YMIN=0; XMAX=279; YMAX=191;(*dimensions of graphics screen*)
TYPE
   CHARSET=SET OF CHAR;
   SHORTSTR=STRING[8];
   BYTE=0..255;
   MEMLOC=PACKED ARRAY [0..1] OF BYTE;
   ACCESS= RECORD
              CASE BOOLEAN OF
                   TRUE:  (ADDRESS:INTEGER);
                   FALSE: (POINTER: ^MEMLOC);
              END;
VAR
   RET: CHAR;

PROCEDURE BEEP;
PROCEDURE SHIFTUP(VAR CH:CHAR);
PROCEDURE GETACHAR(VAR CH:CHAR; LEGALSET:CHARSET);
PROCEDURE GETTEXTCHAR(X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET);
PROCEDURE GETRESPONSE(X,Y:INTEGER; VAR S:SHORTSTR; MAXLEN:INTEGER;
                      LEGALSET:CHARSET);
PROCEDURE GETHICHAR(X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET);
```

```
PROCEDURE GETHIRESPONSE(INITX,Y:INTEGER; VAR S:SHORTSTR;
                              MAXLEN:INTEGER; LEGALSET:CHARSET);
PROCEDURE WSTAT(X,Y:INTEGER; S:STRING);
PROCEDURE FILLBOX(LEFT,RIGHT,BOTTOM,TOP: INTEGER; COLOUR;SCREENCOLOR);
PROCEDURE MOVECOL(X,Y:INTEGER; COL:SCREENCOLOR);
PROCEDURE DRAWLINE(X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);
FUNCTION AROW (NUM:INTEGER; CH:CHAR):CHAR;
FUNCTION AT(X,Y: INTEGER): CHAR;
PROCEDURE WAIT(DELAY:INTEGER);
FUNCTION PEEK(ADDRES: INTEGER):BYTE;
FUNCTION KEYIN:BOOLEAN;
```

Implementation section

A complete listing is given in Appendix B. A description of procedures and functions in this unit is presented here:

PROCEDURE BEEP

BEEP sounds the computer's bell by writing an ASCII 07 (chr(7)).

PROCEDURE SHIFTUP(VAR CH:CHAR)

If variable CH, contains a lower-case alphabetical character, then SHIFTUP will convert this character to upper-case.

PROCEDURE GETACHAR (VAR ACH: CHAR; LEGALSET: CHARSET)

GETACHAR returns a character, ACH. Only a character in specified set LEGALSET will be accepted. The character is not echoed to the screen and the return key is not required to terminate input.

PROCEDURE GETTEXTCHAR (X,Y: INTEGER; VAR ACH:CHAR;
LEGALSET: CHARSET)

GETTEXTCHAR returns a character, ACH. GETTEXTCHAR accepts a character input from the keyboard. Only a character in specified set, LEGALSET will be accepted and echoed to the text screen at position (X,Y). The character may be corrected with backarrow key and input must be terminated with the return key.

PROCEDURE GETRESPONSE (X,Y: INTEGER; VAR S: SHORTSTR;
MAXLEN: INTEGER; LEGALSET: CHARSET)

GETRESPONSE returns a short string, S. GETRESPONSE accepts up to a maximum number of characters, MAXLEN, from the keyboard. Only characters in a specified set, LEGALSET, will be accepted and echoed to the text screen at a specified screen position (X,Y). Input may be corrected using the backarrow key.

PROCEDURE GETHICHAR (X,Y: INTEGER; VAR ACH: CHAR;
LEGALSET: CHARSET)

GETHICHAR is identical to GETTEXCHAR except that the character is echoed to the hi-resolution screen instead of the text screen.

PROCEDURE GETHIRESPONSE (X,Y: INTEGER; VAR S:SHORTSTR;
MAXLEN:INTEGER; LEGALSET: CHARSET)

GETHIRESPONSE is identical to PROCEDURE GETRESPONSE except that accepted input is echoed to hi-resolution screen instead of the text screen.

PROCEDURE WSTAT (X,Y : INTEGER; S: STRING)

WSTAT moves the "turtle" to coordinate (X,Y) on hi-resolution screen and displays string, S.

PROCEDURE FILLBOX (LEFT,RIGHT,BOTTOM,TOP:INTEGER;
COLOUR:SCREENCOLOR)

FILLBOX paints the rectangular area borded by LEFT,RIGHT,BOTTOM and TOP with the COLOUR specified.

PROCEDURE MOVECOL (X,Y: INTEGER; COL: SCREENCOLOR)

MOVECOL moves the "turtle" to coordinate (X,Y) on hi-resolution screen and sets pencolor to COL.

PROCEDURE DRAWLINE (X1,Y1,X2,Y2: INTEGER;
COL:SCREENCOLOR)

DRAWLINE draws a line on hi-resolution screen between points (X1,Y1) and (X2,Y2) in specified colour, COL.

FUNCTION AROW (NUM: INTEGER; CH:CHAR):CHAR)

AROW draws a row of specified characters, CH, on text screen starting at current cursor position. NUM indicates the number of characters to be displayed. AROW is designed as a function rather than a procedure so that it may be incorporated into a write statement. AROW returns a dummy null character.

FUNCTION AT (X,Y: INTEGER): CHAR)

AT moves the cursor to text screen coordinate (X,Y). AT has been designed as a function for the same reasons as AROW. AT returns a dummy null character.

PROCEDURE WAIT(TIME: INTEGER)

WAIT causes a pause of length TIME.

FUNCTION KEYIN(:BOOLEAN)

KEYIN returns true if a key has been pressed since the last time the keyboard was read.

2.3 GRAPHICS PROCEDURES

The Turtlegraphics and Useful procedures necessary to produce the graphics (in the three software packages) are briefly discussed.

2.3.1 METHODS FOR DRAWING SHAPES

Shapes are displayed on the screen using two different methods:

1. "Turtle" commands

A number of "turtle" procedures are available for drawing shapes on the graphics screen. Extensive use is made of three "turtle" procedures from the Turtlegraphics unit:

PENCOLOR (which selects colour of pen);

MOVE (which moves the "turtle" a specified distance in current direction);

MOVETO (which moves the "turtle" from current position to specified coordinate).

Procedure MOVECOL in the Useful unit is a combination of the Move and Pencolor procedures.

2. Bit-map transer

Procedure Drawblock in the Turtlegraphics unit produces graphics by bit-map transfer. Any shape to be drawn on the graphics screen may be represented by a two dimensional packed array of boolean. The array is initialized so that each true bit corresponds on the graphics screen to a pixel turned on and each false bit corresponds to a pixel turned off. The Drawblock procedure is used to copy the array of Boolean to the graphics screen.

The Drawblock declaration is:

```
PROCEDURE DRAWBLOCK(SOURCE: BOOLEAN-ARRAY-TYPE; SIZE, XSKIP, YSKIP,
                    WIDTH, HEIGHT, XSCREEN, YSCREEN, MODE: INTEGER);
```

The parameter list is briefly described as follows:

SOURCE	-	any packed array of boolean;
SIZE	-	bytes per row of source;
XSKIP	-	bits to skip horizontally;
YSKIP	-	bits to skip vertically;
WIDTH	-	bits to plot from xskip to xskip+width;
HEIGHT	-	bits to plot from yskip to yskip+height;
XSCREEN	-	x coordinate to begin plotting;
YSCREEN	-	y coordinate to begin plotting;
MODE	-	display mode;

There are sixteen display modes numbered 0 to 15, with mode number 10 (direct bit copy) being the default mode.

Small, filled-in shapes are most suitable to be drawn using the Drawblock procedure.

2.3.2 METHODS FOR WRITING TEXT ON GRAPHICS SCREEN

Writing text is a specialised case of drawblocking arrays of boolean onto the graphics screen. Apple Pascal provides a file called SYSTEM.CHARSET. This file contains an array representing alphanumeric and special characters which may be displayed on graphics screen. Each character occupies an array 7 pixels wide by 8 pixels high.

The Turtlegraphics unit provides three procedures for writing text to the graphics screen:

WCHAR (writes a character at current screen position);

WSTRING (writes a string at current screen position);

CHARTYPE (defines mode).

Wchar and Wstring procedures use the Drawblock procedure to copy characters from the array defined by SYSTEM.CHARSET to the graphics screen.

Procedure WSTAT from the Useful unit, which writes a string at a specified screen coordinate, is a combination of the Moveto and Wstring procedures.

2.3.3 ANIMATION

Irrespective of the method use to display shape on the screen, movement is created by erasing the shape at current position and then re-drawing it at a different position.

Methods of erasing shapes on the graphics screen:

(i) Shapes drawn with "turtle" commands may be erased by re-drawing the exact shape at the same position using the following colours:

- black (if original shape was drawn in white);
- black1 (if original shape in white2, green or violet);
- black2 (if original shape in white2, orange or blue)

This method is only suitable for simple shapes which may be drawn quickly.

(ii) Shapes created by the drawblock procedure may be erased by drawblocking at the same position using particular character modes. Character mode number six, the Exclusive Or mode (XOR) is the most useful. The XOR logic combines

bits according to the following truth table:

Current screen bit	New bit	XOR combination
F (pixel off)	F (pixel off)	F (pixel off)
T (pixel on)	F (pixel off)	T (pixel on)
F (pixel off)	T (pixel on)	T (pixel on)
T (pixel on)	T (pixel on)	F (pixel off)

If both bits are on, the result is a blank bit on the screen. Therefore if a figure is redisplayed at the same position, it will be erased. Only the figure in the new bit will be erased, without disturbing any of the other display information already present on the screen.

(iii) A section of the graphics screen may be quickly erased by using two Turtlegraphics procedures, Viewport and Fillscreen:

VIEWPORT (creates a "window" on graphics screen);

FILLSCREEN (fills current "window" with specified colour).

Therefore, a rectangular section of the screen is erased by setting the "window" to the appropriate boundaries with the Viewport procedure; erasing everything within this "window" by using Fillscreen specifying a black colour; and finally the "window" is reset to entire graphics screen. Procedure FILLBOX, is a combination of Viewport and Fillscreen procedures and enables a section of the screen to be painted a specified colour.

CHAPTER 3: ACID/BASE TITRATION PACKAGE

3.1 DESCRIPTION OF ACID/BASE TITRATION PROGRAMS

3.1.1 User interface for package

3.1.2 Titration Features

3.1.3 Titration Program

3.1.4 Indicators Program

3.1.5 Quiz Program

3.1.6 Assignment Program

3.2 OBJECTIVES OF THE TITRATION PACKAGE

3.2.1 Worksheets

3.3 PASCAL CODE FOR TITRATION PACKAGE

3.3.1 Titration Library Unit

3.3.2 Production of pH graph

3.3.3 Calculation of pH

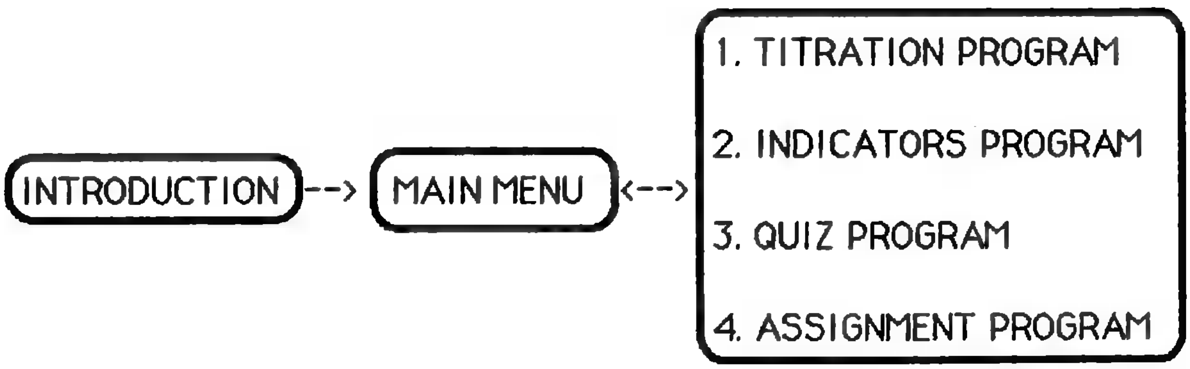
3.1 DESCRIPTION OF ACID/BASE TITRATION PACKAGE

The ACID/BASE TITRATION PACKAGE consists of four programs each of which allows the user to specify certain conditions before carrying out a simulated titration. The titration may be between any monoprotic or diprotic acid and any monoprotic base. A general description of the package is given followed by a detailed discussion of the individual programs.

3.1.1 USER INTERFACE FOR ACID/BASE TITRATION PACKAGE

The ACID/BASE TITRATION PACKAGE appears to the user in three main sections:

- an introduction section
- a main menu
- four programs each with optional instruction program. Each of these four programs links back to main menu.



The ACID/BASE TITRATION PACKAGE containing the above programs is located on one disk. This disk must always be resident in the disk drive as it is necessary to access the disk each time the program moves from one program to another.

Introduction to package

The introduction consists of four pages on the graphics screen. These introductory pages are only encountered when the disk is first booted.

Page 1. Title page.

Page 2. Informs user that all input must be followed by the RETURN key.

Page 3. Asks user whether a colour monitor is being used. This information is stored in system memory to be accessed later by each of the four programs. The graphics is slightly different in each program depending on whether a colour monitor is available.

Page 4. Informs user that entering "Q" is required to exit from programs. Entering "Q" at any interrogative point in any of the four programs, will exit the user from the current section of the program. Depending on the position in the program, the user will usually be given the option of carrying out another titration in the same program or going back to the main menu. Therefore if the user wishes to change some input after it has been entered, entering "Q" at the next prompt for input will allow the user to reenter input for that particular titration.

Main menu

This package is menu driven, with the following main menu:

Titration of Acids & Bases.	..... (1)
Titration of Acids & Bases Using Indicators.	..... (2)
Titration Quiz.	..... (3)
Titration Assignment.	..... (4)
Quit	..... (Q)

Four titration programs

After selecting one of the four titration programs, the user is given the option of viewing instructions regarding the program selected. These instructions may be repeated as many times as required, and the user may exit at any point in the instructions by entering "Q".

3.1.2 TITRATION FEATURES

The following features are common to all titration programs:

1. Titration conditions

The user is able to specify a number of conditions, such as the concentration of the acidic and basic solutions, involved in the titration. The task of defining the titration conditions is very simple - the user is presented with a series of options and in most cases it is only necessary to input a numeral or letter corresponding to a desired option.

2. Screen layout

Throughout the titration, experimental parameters specified by the user, such as concentration of acid and base, are displayed on the screen. The graphic illustrates a flask and the lower section of a burette. As the titration procedes the volume of titrant currently in the flask and pH of the solution is calculated and updated on the screen. Figure 3.1 illustrates a typical screen layout for a titration.

The *Titration* and *Indicators* programs in the package use the right hand side of the screen to plot the pH against the volume of titrant as the titration is carried out. The *Quiz* and *Assignment* programs simply use the right hand side of the screen to specify the current titration.

3. Titration procedures

The space bar is used to release an increment of titrant from the burette. This volume of titrant increment is set by the user at the start of the titration but may be changed at any time during the titration. Three keys are available to the user during a

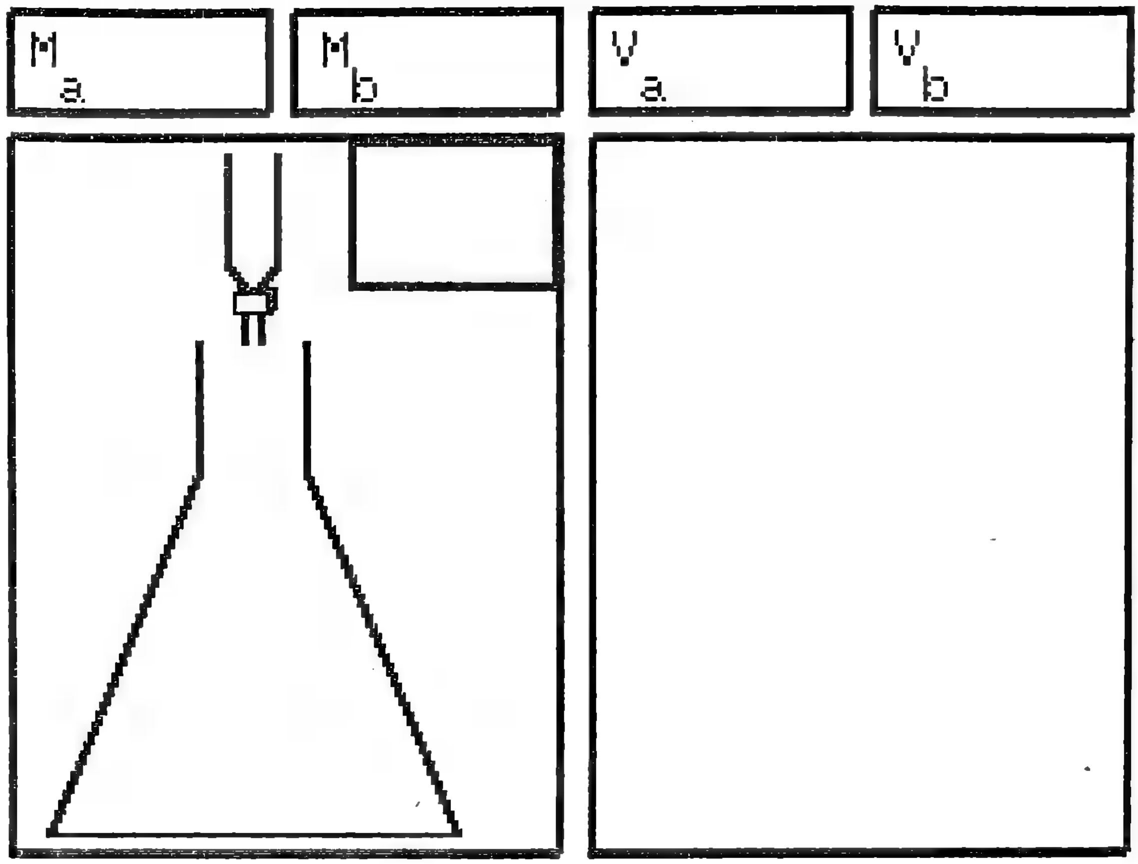


Figure 3.1 Basic screen layout for all titration programs.

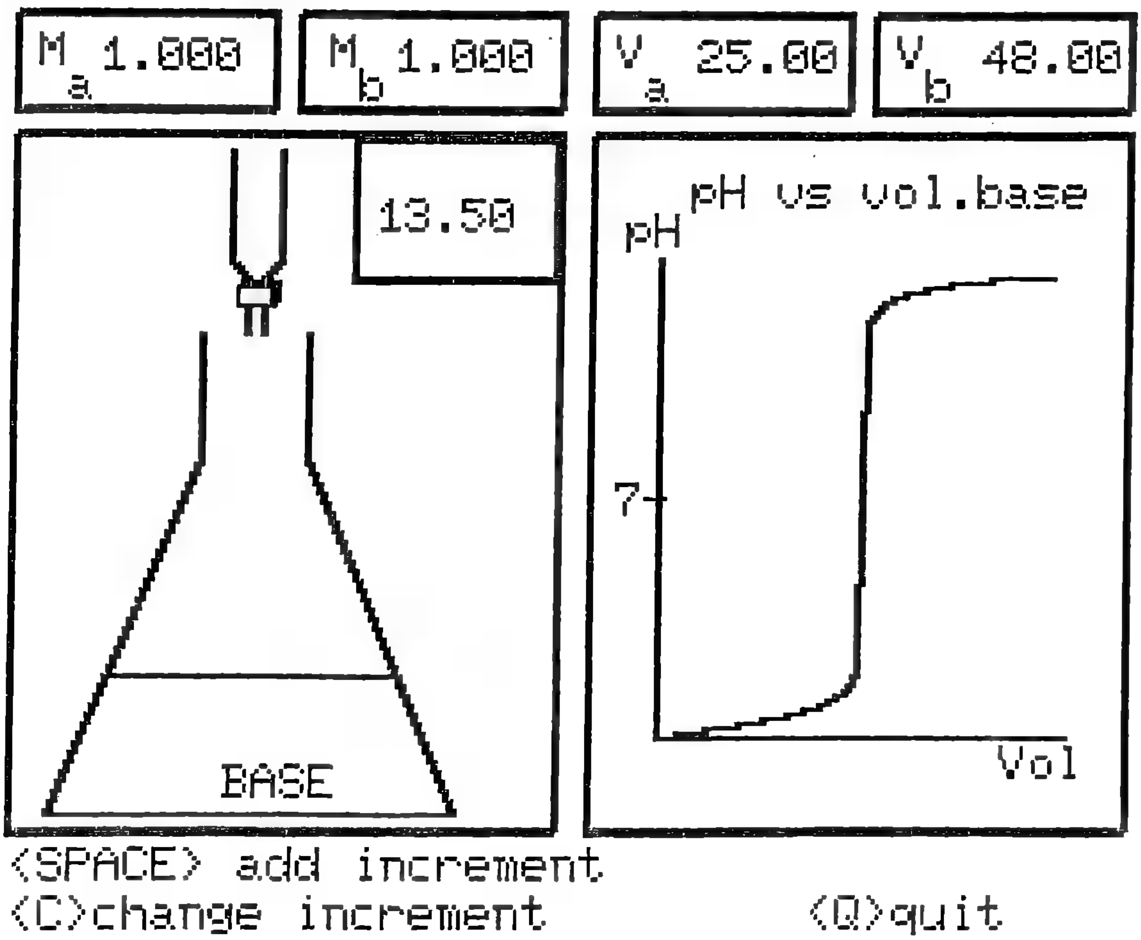


Figure 3.2 Typical screen layout for Titration Program.

titration:

(i) <SPACE BAR> - to add an increment of titrant to flask.

(ii) < C > - to change volume of titrant increment

(iii) < Q > - to quit from titration.

4. Titration graphics

When the space bar is pressed the following animation and data appear on the screen:

(i) An increment of titrant is released from the burette and falls into the flask. The volume of solution in the flask increases.

(ii) Total volume of titrant added to the flask is updated.

(iii) pH of the solution is updated.

(iv) If the end point has been reached (or overshot) the colour of the solution in the flask changes.

5. Repeat options

The titration may be continued (by repeatedly pressing SPACE BAR) until either "Q" is pressed, indicating that the user wishes to quit from the titration, or until the flask becomes full. When the flask is full the user is forced to quit from the titration. After quitting from a titration the user is presented with a number of options. The variety of options is different for each of the four programs, however all programs offer at least the following three options:

(i) Repeat the previous titration.
If a titration is repeated it is not necessary to respecify the titration conditions.

(ii) Carry out a different titration.

(iii) Quit from the current acid/base program. This takes the

user back to the main menu for this package.

3.1.3 TITRATION PROGRAM

This program allows the user to carry out a simulated titration between a specified acid and base. The solution contains an ideal indicator which causes a colour change at the equivalence point. Throughout the titration the pH of the solution is displayed and plotted against volume of titrant.

HOW TO SPECIFY A PARTICULAR TITRATION

The following options are available:

Option 1. Type of titration

The main menu for this program offers the following choices:

1. strong acid/strong base titration
2. weak acid/ strong base titration
3. weak base/strong acid titration
4. diprotic acid/strong base titration

Option 2. Strength of acid and base

The strength of acids and bases presented depends on the type of titration chosen in option 1. One menu will provide a selection of acids, and a second menu will provide a selection of bases. One of the following menus will be displayed for selection of acid:

Strong acids (displayed if "1" or "3" was chosen in option 1):

1. hydrochloric acid
2. nitric acid
3. perchloric acid

Weak acids (displayed if "2" was chosen in option 1):

1. acetic acid

2. hydrocyanic acid
3. hydrofluoric acid
4. benzoic acid
5. input pKa

Diprotic acids (displayed if "4" was chosen in option 1)

1. sulfuric acid
2. carbonic acid
3. oxalic acid
4. tartaric acid
5. input pK1 & pK2

One of the following menus will be displayed for selection of base:

Strong bases (displayed if "1" ,"2" or "4" was chosen in option 1):

1. sodium hydroxide
2. potassium hydroxide

Weak bases (displayed if "3" was chosen in option1):

1. ammonia
2. pyridine
3. input pKb

pK values

The user may select an unlisted acid or base by entering its pK value(s). The pKa and pKb values for monoprotic acids and bases are restricted to the range of 1 to 11. For diprotic acids pK1 must be in range 1 to 11 and pK2 must be less than 13. Furthermore pK1 must be less than pK2. The program will not accept values which do not fall within these ranges.

Option 3. Concentration of acid and base

The user must enter the concentration of the acid and base which were chosen in option 2. Concentrations must be in the range 0.001 molar to 1.000 molar. These values are displayed on the screen throughout the titration.

Option 4. Titrant and titrand

The user must determine which solution is to act as the titrant (i.e. the solution in the burette) and which solution is to act as the titrand (i.e. the solution in the flask). Having decided upon which solution is to be the titrand it is necessary to decide on the volume of this solution. The volume of the titrand must be in the range of 10.00mL to 50.00mL. The volume of the titrand and titrant are displayed on the screen during the titration.

Option 5. Labelling of solution

The user is given the option of having the solution in the flask identified with a label. If this option is selected then the following labels are used for titrations involving monoprotic acids:

(i) At the start of the titration the solution is labelled "acid" or "base" depending on the nature of the titrand.

(ii) At the equivalence point the solution is labelled "end pt".

(iii) At other points in the titration the solution is labelled "acid", "base" or "buffer" depending on the nature of the solution.

For titrations involving diprotic acids the following labels are used:

(i) At the start of the titration the solution is labelled "diacid". In this program the diprotic acid always acts as the titrand.

(ii) At the first and second equivalence points the solution is labelled "endpt1" and "endpt2" respectively.

(iii) Before the first equivalence point the solution is labelled "buffer".

(iv) Between the first and second equivalence points the solution is labelled "2salts".

(iv) After the second equivalence point the solution is labelled "base".

After the above options have been specified , thereby determining the conditions for a titration, the simulated titration is ready to commence.

SIMULATED TITRATION FOR TITRATION PROGRAM

1. Screen layout

Throughout the simulated titration the following information is displayed on the screen.

(i) concentration of acid
(ii) concentration of base
(iii) volume of acid in flask
(iv) volume of base in flask
(v) pH of solution in flask
(vi) graph of pH vs. volume of titrant

Figure 3.2 illustrates the typical screen layout for titration simulations in the *Titration* program.

2. Titration graphics

In addition to the features common to all the titration programs,

(Section 3.1.2), the following information is displayed on the screen during the titration:

(i) As the titration progresses a graph of pH vs. volume of titrant is produced. As each increment of titrant is added (using space bar), the graph of pH is extended to cover the volume of titrant added in the last increment. If a different titration is carried out then the user is given the option of retaining the current graph. In this way a series of superimposed curves may be obtained on the same graph. (Figure 3.3)

(ii) The solution contains an ideal indicator which changes colour at the equivalence point of the titration. Therefore the end point and equivalence point of these titrations are identical.

(iii) This program allows for the nature of the solution to be labelled. After each addition of titrant, the program alters the label if necessary. The end point of the titration is identified by the label "end pt". This feature enables the user to confirm when the end point has been reached or overshot.

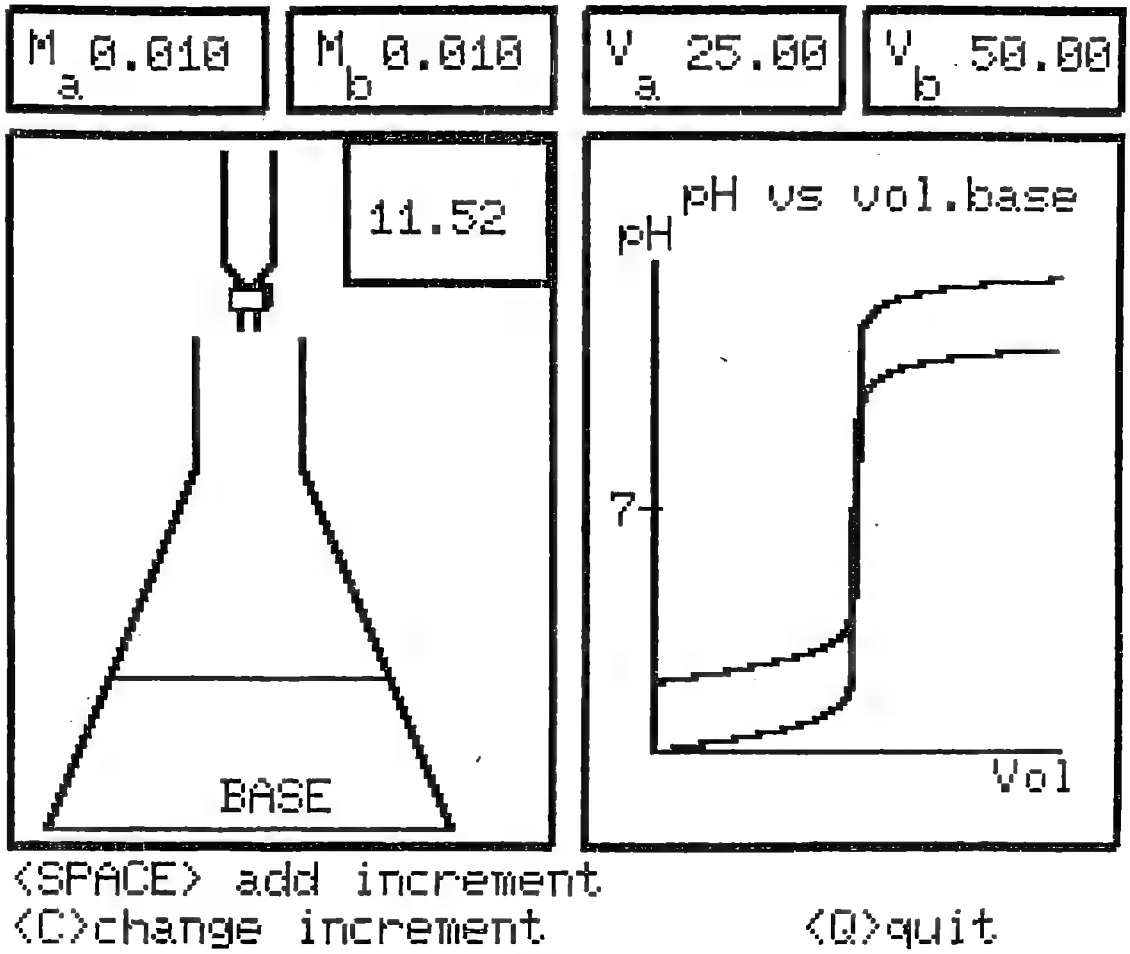


Figure 3.3 Titration Program: a series of superimposed pH curves drawn on graph.

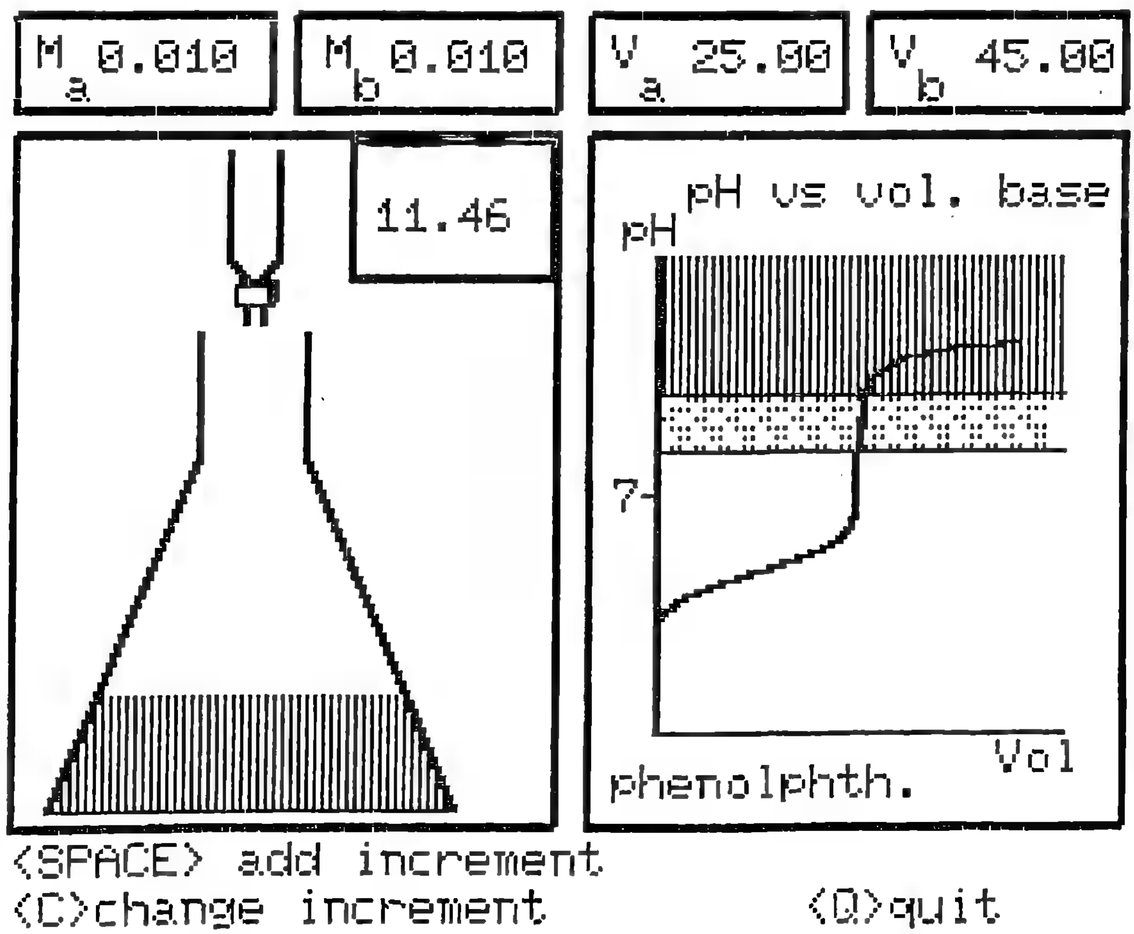


Figure 3.4 Indicator Program: pH colour ranges for selected indicator is displayed on graph.

3.1.4 INDICATORS PROGRAM

The major difference between the *Indicators* program and the *Titration* program is that the user may select a variety of common acid/base indicators.

HOW TO SPECIFY A PARTICULAR TITRATION

As in the *Titration* program, the user must specify the conditions for each titration. The following options are available in the *Indicators* program:

Option 1. Type of titration.

Option 2. Strength of acid and base.

Option 3. Concentration of acid and base.

Option 4. Titrant and titrand.

Whereas the *Titration* program provided an ideal indicator for which the endpoint coincided with the equivalence point, this program offers a choice of indicators. The indicators available are:

- methyl orange
- methyl red
- litmus
- bromothymol blue
- phenolphthalein
- thymolphthalein
- ideal

The option of labelling the solution in the flask (Option 5 in the *Titration* program) is not provided.

SIMULATED TITRATION FOR INDICATORS PROGRAM

1. Screen layout and graphics

The screen layout and graphics are almost identical to those in the *Titration* program. An additional feature displayed on the pH graph is the range in which the selected indicator exists in particular colours. The name of the indicator chosen for a particular titration is displayed beneath the pH graph. (Figure 3.4) The solution changes colour at a pH determined by the indicator chosen rather than the pH at the equivalence point. The graphics in this program are most effective on a colour monitor. Each indicator changes colour over a pH range. If this pH range lies on the steep part of the titration curve, the indicator colour will change directly from one colour to another colour. However, if the pH range does not lie within the steep part of the curve then the indicator passes through stages where the colour gradually changes from one colour to another.

These graphics illustrate many aspects of acid/base indicators, such as the difference between the end point and equivalence point of a titration.

2. Repeat options

In addition to the standard repeat options (Section 3.1.2), the user is given the option of repeating the last titration using a different indicator. By selecting this option a comparison between the various indicators can readily be made.

3.1.5 QUIZ PROGRAM

The *Quiz* program provides an acidic or basic solution and the user must carry out a series of titrations in order to determine its concentration. Throughout the titration the pH of the solution is displayed and an ideal indicator changes colour at the equivalence point.

The concentration of each quiz solution is randomly selected by the program. This program allows the user to practice acid/base titration methods and associated calculations. After carrying out a titration, the user is required to enter the concentration of the quiz solution and will then be informed as to the accuracy of this result.

HOW TO OBTAIN A QUIZ SOLUTION

The user has the option of selecting the type of quiz solution from the following list:

1. hydrochloric acid
2. sodium hydroxide
3. acetic acid
4. ammonia
5. oxalic acid

HOW TO SPECIFY A PARTICULAR TITRATION

If the quiz solution is acidic, then the standard solution will be sodium hydroxide, otherwise the solution is basic and the standard solution is hydrochloric acid. Furthermore, the program determines which solution will be the titrant and titrand.

The user has only two options for specifying the conditions of the titration:

Option 1. Concentration of standard solution

The concentration of the standard solution may be determined in two ways:

(i) The computer will determine the most suitable concentration.

(ii) The user selects from a list of available concentrations: 0.010M, 0.05M, 0.100M, 0.500M and 1.00M.

Option 3. Volume of titrand

It is necessary to select the volume of the titrand which must be in the range of 10.00mL to 50.00mL.

SIMULATED TITRATION FOR QUIZ

1. Screen layout and graphics

The screen layout and graphics are similar to the *Titration* and *Indicators* program except that a graph of pH vs volume of titrant is not given. Throughout the titration the following information is displayed on the screen

(i) concentration of known solution

(ii) volume of titrant.

(iii) volume of titrand.

(iv) pH of solution in flask.

2. Repeat options

After quitting from a titration (by entering "Q") the user is given the following options:

(i) Repeat the previous titration.

(ii) Repeat the titration but alter the conditions.

(iii) Enter concentration of unknown.

(iv) Get calculator.

(v) Give up.

(vi) Quit from the quiz program.

Typically, the student would repeat the titration several times, selecting options (i) or (ii) above, until the end point had been established. If option (ii) is selected the concentration of the standard solution and/or the volume of the titrand may be altered. After completing the titrations the student must calculate the concentration of the quiz solution. For this purpose, a calculator option is included which handles the operations of addition, subtraction, multiplication and division.

When the concentration has been calculated, this value is entered, (option (iii)) and the program indicates the accuracy of this value. Depending on the percentage error, one of the following comments is made:

relative error	comment
< 1%	excellent
1-3%	very close
4-6%	close
> 6%	not too good
> 20 %	terrible

In addition to one of the above comments, the user is informed of the correct concentration of the quiz solution. If the user has been unable to calculate the concentration of the solution, the "give up" option simply provides the correct concentration.

After entering the concentration, or giving up, the user is given the choice of attempting another quiz solution or quitting from this program. Quitting will take the user back to the main menu for this package.

3.1.6 ASSIGNMENT PROGRAM

The *Assignment* program is very similar to the *Quiz* program in that the user must determine the concentration of an acidic or basic solution. However this program does not allow the user to enter an estimation of the concentration nor does it inform the user of the correct concentration. The *Assignment* program provides one hundred solutions of different concentration. The student must carry out a series of titrations in order to determine the concentration of the assignment solution.

HOW TO OBTAIN AN ASSIGNMENT SOLUTION

The student must enter an assignment number between 0 and 99, which is allocated by the teacher. Each assignment number corresponds to a different assignment solution. The key to these assignment numbers is provided in the instruction manual for this package, listed in Appendix A. The program uses a mathematical formula to convert the assignment number into the concentration of the assignment solution. The concentration is in the range of 0.100 molar to 1.000 molar. The assignment number also determines the nature of the assignment solution.

HOW TO SPECIFY A PARTICULAR TITRATION

After entering the assignment number, the user has only two options available for specifying a titration:

Option 1. Concentration of standard solution

The nature of the standard solution is determined by the program however the user must determine the concentration of this solution. The concentration must be in the range of 0.100 molar to 1.000 molar.

Option 2. Volume of titrand

The program determines which solution is to be the titrand, however the user must specify the volume of titrand. This volume must be in range of 10.00mL to 50.00mL.

SIMULATED TITRATION FOR ASSIGNMENT

The simulated titration is identical to the *Quiz* titration. After quitting from the titration, the user is given the following options:

(i) Repeat the previous titration.

(ii) Repeat titration but altering conditions.

(iii) Get calculator.

(iv) New assignment.

(v) Quit from the assignment program.

The *Assignment* program is designed so that a teacher may allocate titration assignments to students. Each student may be given the task of solving a series of assigments distinguished by different assignment numbers. After carrying out a series of assignment titrations, the student completes a written report (worksheet number 10, listed in Appendix A, provides a suggested format for such a report) noting the titration data and calculations required to determine the concentration of the allocated assignment solution(s). The teacher evaluates the assignments by referring to the concentration corresponding to each assignment number, listed in the instruction manual for this package.

3.2 OBJECTIVES OF THE ACID/BASE TITRATION PACKAGE

This titration package can be used in a variety of ways. The programs may be used by the teacher as visual aids or by students as an investigative tool. Many chemical concepts which students often find difficult are readily illustrated and explained by these programs.

The four programs should be viewed in the order in which they appear in the main menu. The objectives of the first two titration programs are to introduce and illustrate various titration concepts. The objectives of the *Quiz* and *Assignment* programs are to consolidate and interrelate these concepts.

1. Titration Program

The *Titration* program has been designed as an aid in introducing the fundamental concepts of acid/base titrations. It contains the following features:

- the use of an ideal indicator which avoids confusion between end point and equivalence point.
- the option to have the nature of the solution labelled provides more information.
- the option of obtaining a graph showing a series of superimposed curves.

This program can be used to investigate how solution variables, such as strength and concentration, affect the reaction between an acidic and basic solution. A comparison between the titrations can readily be carried out.

2. Indicators Program

This program builds on the fundamental titration concepts to illustrate the criteria necessary for an indicator to be suitable

for a particular titration. Using this program it is very simple to demonstrate why an indicator may be suitable for one titration but not for another.

The fact that the range in which the indicator changes colour does not necessarily include the pH of the solution at the equivalence point, is often very confusing for students. For example, a titration between hydrochloric acid and sodium hydroxide has a pH of 7.0 at the equivalence point. This program clearly demonstrates why phenolphthalein, which changes colour in range 8.3 - 10.0, is a suitable indicator for this reaction.

3. Quiz and Assignment Programs

In these two programs the student must apply the concepts introduced in the earlier programs in order to determine the concentration of a sample solution. The *Quiz* program provides the option that the user may allow the program to select the most appropriate concentration for the standard solution. This feature is not available in the *Assignment* program.

The *Quiz* and *Assignment* programs are designed for student use rather than teacher demonstration. The *Assignment* program may be used for student assessment.

3.2.1 WORKSHEETS

A series of worksheets has been designed for use with these programs. These worksheets guide the student through the investigation of concepts relating to titrations. Each worksheet requires a series of titrations to be carried out. Within each series of titrations only one condition is varied so the effect of

this variable on the titration is observed.

<u>Worksheet 1.</u> Each titration in this series involves 1.00 molar sodium hydroxide solution reacting with a 1.00 molar solution of strong acid. A different strong acid is used in each titration.

<u>Worksheet 2.</u> This series involves potassium hydroxide solution reacting with hydrochloric acid. In each titration the concentration of the acid is the same as the concentration of the base, however these concentrations are different in each titration. In the first titration, both acid and base are 1.00 molar solutions, in the second titration both are 0.10 molar solutions, etc.

<u>Worksheet 3.</u> This series of titrations involves sodium hydroxide solution reacting with nitric acid. In each titration the concentration of base is held constant whilst the concentration of acid is altered.

<u>Worksheet 4.</u> The previous worksheet involved only strong acids and bases. This worksheet involves titrations between acetic acid, a weak acid, and potassium hydroxide, a strong base. In each titration the concentration of acetic acid is the same as the concentration of potassium hydroxide solution, however this concentration varies between each titration.

<u>Worksheet 5.</u> This series of titrations involves acids of varying strength reacting with a strong base. In each

titration 1.00 molar sodium hydroxide in reacted with a 1.00 molar solution of weak acid.

<u>Worksheet 6.</u> This series of titration involves ammonia reacting with hydrochloric acid. In each titration the concentration of the acid is held constant whilst the concentration of base is altered.

<u>Worksheet 7.</u> This worksheet involves titrations between sodium hydroxide solution and various diprotic acids. In each titration 1.00 molar sodium hydroxide is titrated with 1.00 molar solution of diprotic acid.

<u>Worksheet 8.</u> This worksheet involves a series of titrations in which a variety of indicators must be used. 1.00 molar hydrochloric acid is reacted with 1.00 molar sodium hydroxide solution. This titration is repeated with both solutions having a 0.01 molar concentration. A third titration reacts 1.00 molar acetic acid with 1.00 molar sodium hydroxide.

<u>Worksheet 9.</u> This worksheet involves a series of titrations for which a suitable indicator must be found.

<u>Worksheet 10.</u> Specimen layout for submission of a titration assignment.

3.3 PASCAL CODE FOR ACID/BASE TITRATION PACKAGE

The ACID/BASE TITRATION PACKAGE uses two customised library units, Useful Unit and Titrlib Unit, both of which have been incorporated into System.Library. The Useful Unit is discussed in Section 2.2.3 and the Titrlib Unit is discussed in Section 3.3.1. In addition to the system files, the following object code files are included in the ACID/BASE TITRATION PACKAGE:

System.startup	(code for introduction section)
Menu.code	(code for main menu and explanation)
Titration.code	(code for Titration program)
Indicator.code	(code for Indicators program)
Quiz.code	(code for Quiz program)
Assign.code	(code for Assignment program)

The source code for all the program files is listed in Appendix C. A discussion of procedures required for the production of the pH graph is given in Section 3.3.2, and a discussion of procedures required for the calculation of pH is given in Section 3.3.3.

3.3.1 TITRATION LIBRARY UNIT

TITRLIB is a library unit used by all programs in the ACID/BASE TITRATION PACKAGE.

DECLARATION SECTION

```
UNIT TITRLIB; INTRINSIC CODE 18 DATA 19;
```

The declaration indicates that TITRLIB is an intrinsic unit for which the code occupies operating system segment 18 and the data occupies segment 19.

INTERFACE SECTION

```
INTERFACE

USES TURTLEGRAPHICS, TRANSCEND,USEFUL;
CONST
   FLASKX=60; FLASKY   (*co-ordinates of centre base of flask*)
   FLASKSIZ=100;       (*size of flask *)
   KW=1.0E-14;         (*ionization constant of water *)

TYPE
   ACIDORBASE=(ACID,BASE);
   TITRAT=(WEAKACID,STRONGACID,WEAKBASE,DIPROTIC);

VAR
   TITRTYPE:  TITRAT;      (* classification of current titration *)
   FLASKTOP,               (* base of neck of flask *)
   NECKTOP:   INTEGER;     (* very top of neck of flask *)
   K1,K2,                  (* ionization constants *)
   FLASKVOL,               (* volume of soln in flask *)
   HCONC, OHCONC:   REAL;(* conc. of acid & base *)
   FILLRATE:  INTEGER;     (* rate at which flask filled*)
   INFLASK:   ACIDORBASE; (* type of soln in flask *)
   QUIT:   BOOLEAN;
   NUMS:   CHARSET;

FUNCTION RVALUE (VAR S:STRING): REAL;
PROCEDURE REALSTR (VAR REALNUM:REAL; VAR  WORD: STRING;
                              DECPOINT,SIZE : INTEGER);
PROCEDURE INRANGERESPONSE (VAR VALUE: REAL; VAR S:SHORTSTR;
                              MIN, MAX : REAL;  X,Y: INTEGER);
PROCEDURE DOUBLELINE (X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);
PROCEDURE DRAWBOX(X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);
PROCEDURE INITSCREEN;
PROCEDURE DRAWAXES (X,Y,SIZE : INTEGER; COL: SCREENCOLOR);
PROCEDURE DRAWFLASK(X,Y,SIZE : INTEGER; COL: SCREENCOLOR);
PROCEDURE FILLFLASK (VAR LTSIDE,RTSIDE,OLDLEVL,INCREASE: INTEGER;
                          COL: SCREENCOLOR);
PROCEDURE MOVEDROP (INCR: REAL; VAR LIQLEVEL: INTEGER);
PROCEDURE ACIDMOLARITY (S: SHORTSTR);
PROCEDURE BASEMOLARITY (S: SHORTSTR);
PROCEDURE ACIDISP (S: SHORTSTR);
PROCEDURE BASEDISP (S: SHORTSTR);
PROCEDURE DISPLAYPH (S: SHORTSTR);
PROCEDURE TWOPROMPTS (S1,S2: STRING);
PROCEDURE SETUPCONDITIONS (VAR HCONC, OHCONC : REAL;
                           VAR INFLASK : ACIDORBASE);
PROCEDURE CHECKKEY (VAR SPACEPR, SELECTCHANGE: BOOLEAN);
PROCEDURE REQUEST;
PROCEDURE INCRPROMPT;
PROCEDURE SELECTINR(VAR   INCR: REAL);
PROCEDURE CHANGEINC (VAR CHANG: BOOLEAN; VAR INCR: REAL);
PROCEDURE CLEARVALUES(VAR NEWCONC:BOOLEAN);
```

```
PROCEDURE BACKTOMENU;
PROCEDURE GETK(VAR K1,K2:REAL);
PROCEDURE SELECTTYPE(VAR TITRTYPE:TITRAT);
PROCEDURE CALCPH (ANYACID, ANYBASE, HCONC, OHCONC: REAL;
                  VAR PH: REAL);
```

IMPLEMENTATION SECTION

A complete listing is given in Appendix C. A description of the procedures in this unit is presented :

FUNCTION RVALUE(VAR S: STRING):REAL;

RVALUE converts a string, S, into a real number.

PROCEDURE REALSTR (REALNUM: REAL; VAR WORD:STRING; DECPOINT, SIZE: INTEGER);

Accepts a real number, REALNUM, and converts it into a string, WORD. This string displays the original number rounded off to the specified number of decimal points, DECPOINT, and the string is padded with leading spaces until it is of length SIZE.

PROCEDURE INRANGERESPONSE (VAR VALUE:REAL;VAR S:SHORTSTR; MIN,MAX: REAL; X,Y: INTEGER);

INRANGERESPONSE returns a real number, VALUE, and its short string representation, S. Input is read as characters and then converted into a real number. The real number is checked to see if it falls within the range specified, MIN to MAX. Only numerals or the letter "Q" are accepted as input and echoed to the hi-resolution screen at position (X,Y). Real numbers must be entered as decimals - scientific notation is not accepted. Input may be corrected with backarrow, and is erased from screen after the return key has been pressed. INRANGERESPONSE repeats itself until a number within the specified range, or the letter "Q", is entered. The letter "Q" is accepted as input so that the user may exit immediately from any point within the program.

PROCEDURE DOUBLELINE (X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);

DOUBLELINE draws a line, two pixels wide, between coordinates (X1,Y1) and (X2,Y2), in specified colour, COL.

PROCEDURE DRAWBOX(X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);

DRAWBOX draws a box having coordinates (X1,Y1) and (X2,Y2) as bottom left hand and top right hand coordinates respectively. The box is drawn with doublelines.

PROCEDURE INITSCREEN;

Draws the two large and four small boxes on the graphics screen as well as the lower end of a burette in the large left-hand box.

PROCEDURE DRAWAXES(X,Y,SIZE: INTEGER; COL:SCREENCOLOR);

DRAWAXES draws x- and y-axes on graphics screen, both of length SIZE, with origin at coordinate (X,Y). The x-axis is labelled "VOL" and the y-axis is labelled "pH". The value "7" is marked off half way up y-axis.

PROCEDURE DRAWFLASK(X,Y,SIZE: INTEGER; COL:SCREENCOLOR);

Draws a conical flask on graphics screen. The flask is has width and height of specified number of pixels, SIZE, and the centre of the base of the flask is at (X,Y).

PROCEDURE FILLFLASK(VAR LTSIDE,RTSIDE,OLDLEVEL,
INCREASE: INTEGER; COL:SCREENCLOR);

FILLFLASK increases level of the solution in flask from the OLDLEVEL by a certain INCREASE. LTSIDE and RTSIDE define the current coodinates of the flask on the same level as the surface of the solution.

PROCEDURE MOVEDROP(INCR: REAL; VAR LIQLEVEL: INTEGER);

A single drop is released from burette and falls until it reaches the current level of solution in the flask, LIQLEVEL. The size of the drop is determined by INCR.

PROCEDURE ACIDMOLARITY(S:SHORTSTR);

String S, which represents molarity of acid, is displayed on graphics screen at an internally determined coordinate position.

PROCEDURE BASEMOLARITY(S:SHORTSTR);

String S, which represents molarity of base, is displayed on graphics screen at an internally determined coordinate position.

PROCEDURE ACIDVOLUME(S:SHORTSTR);

String S, which represents volume of acid, is displayed on graphics screen at an internally determined coordinate position.

PROCEDURE BASEVOLUME(S:SHORTSTR)

String S, which represents volume of base, is displayed on graphics screen at an internally determined coordinate position.

PROCEDURE DISPLAYPH(S:SHORTSTR)

String S, which represents pH of solution, is displayed on graphics screen at an internally determined coordinate position.

PROCEDURE TWOPROMPTS(S1,S2:STRING);

TWOPROMPTS displays the two strings, S1 and S2, at the bottom of the graphics screen.

PROCEDURE SETUPCONDITIONS(VAR HCONC,OHCONC: REAL;
VAR INFLASK: ACIDORBASE);

The user is prompted to enter concentration of acid and base to be used in titration, HCONC and OHCONC. Only concentrations in range 0.001 to 1.000 molar are accepted. SETUPCONDITIONS asks

user which solution is to be in the titrand, INFLASK, and then prompts user to enter the volume of the titrand, FLASKVOL. All prompts appear on graphics screen. Input may be corrected with a backarrow. Only numerals, or "Q" (to quit) are acceptable input which will be echoed to the graphics screen.

PROCEDURE CHECKKEY (VAR SPACEPR,SELECTCHANGE,QUIT: BOOLEAN);

CHECKKEY monitors keyboard input until one of three characters is entered - the space bar, "C" or "Q" thereby setting the respective Boolean flags - SPACEPR, SELECTCHANGE or QUIT. Input is not echoed to screen.

PROCEDURE REQUEST;

REQUEST displays following prompts on the graphics screen during the titration:
"<SPACE> add increment"; "<C>change increment"; "<Q>quit".

PROCEDURE INCRPROMPT;

INCRPROMPT displays prompt on the graphics screen to select volume of titrant increment.

PROCEDURE SELECTINCR(VAR INCR: REAL);

Prompt to enter increment of titrant is displayed on graphics screen. Value entered must be within range of 0.05 to 10.00 mL. Input may be corrected with a backarrow. Only numerals, or "Q" (to quit) are acceptable input which will be echoed to the graphics screen.

PROCEDURE CHANGEINC (VAR CHANG: BOOLEAN; VAR INCR: REAL);

CHANGEINC erases the prompt lines at bottom of the graphics screen; allows user to select new titrant increment; and finally

replaces prompt lines to be displayed during titration.

PROCEDURE CLEARVALUES(VAR NEWCONC:BOOLEAN);

CLEARVALUES erases the volume of acid and base, pH value and titration flask from the the graphics screen. If the boolean flag, NEWCONC, indicates that the concentration of solutions is to be changed, then the current concentrations are also erased.

PROCEDURE SETCOLOUR;

SETCOLOUR reads message previously stored by the introduction program, and sets boolean flag indicating whether colour monitor is in use.

PROCEDURE BACKTOMENU;

BACKTOMENU displays message on the text screen indicating the main menu is being reloaded into memory.

PROCEDURE GETK(VAR K1,K2: REAL);

GETK prompts user to enter the pK value(s) for the weak acid or base. The pK values are converted into the dissociation constants, K1 and K2.

PROCEDURE SELECTTYPE(VAR TITRTYPE:TITRAT);

SELECTTYPE prompts user to enter the acid and base to be used in titration. If either the acid or base is weak, then the strength may be determined by inputting pK value(s). TITRTYPE is set to indicate the nature of titration based on solutions selected.

PROCEDURE CALCPH (ANYACID, ANYBASE, HCONC, OHCONC: REAL;
VAR PH: REAL);

CALCPH returns pH of the titration solution, being passed the following information:

- Volume of acid (ANYACID).

- Volume of base (ANYBASE).
- Concentration of acid (HCONC).
- Concentration of base (OHCONC).

CALCPH accesses global variables(TITRAT,K1,and K2) to determine the strength of acidic and basic solutions.

3.3.2 PRODUCTION OF THE PH GRAPH

1. Interpolating points.

In order to obtain an accurate pH curve a reasonable number of points must be plotted irrespective of the number of points encountered in the titration. These values are calculated prior to the commencement of the simulated titration and stored in two arrays. One array contains various titrant volumes and the other array the corresponding pH values. During the titration these values are extracted to plot points in between those actually encountered in the titration.

For example, consider a titration between 10mL of 0.01 molar hydrochloric acid and 0.10 molar sodium hydroxide. The initial pH is 2.00 and if three 1 ml aliquots of titrants were added then pH values of 7.0, 12.0 and 12.2 would be observed. If these were the only values used to plot the graph then the pH curve displayed in Figure 3.5 would be obtained. Interpolation of points gives the more accurate pH curve, displayed in Figure 3.6.

2. Scaling horizontal axis.

The volume of titrant added is represented on the horizontal axis of the graph. In these programs the volume has been scaled so as to represent a percentage of the titration. For a monoprotic acid/base titration the horizontal axis represents 200% of the titration (twice the volume required to reach the equivalence point) and for diprotic acid/base titrations the horizontal axis represents 300% of the titration (three times the volume required to reach the first equivalence point). This feature is essential if several graphs are to be superimposed, each having a different equivalence volume.

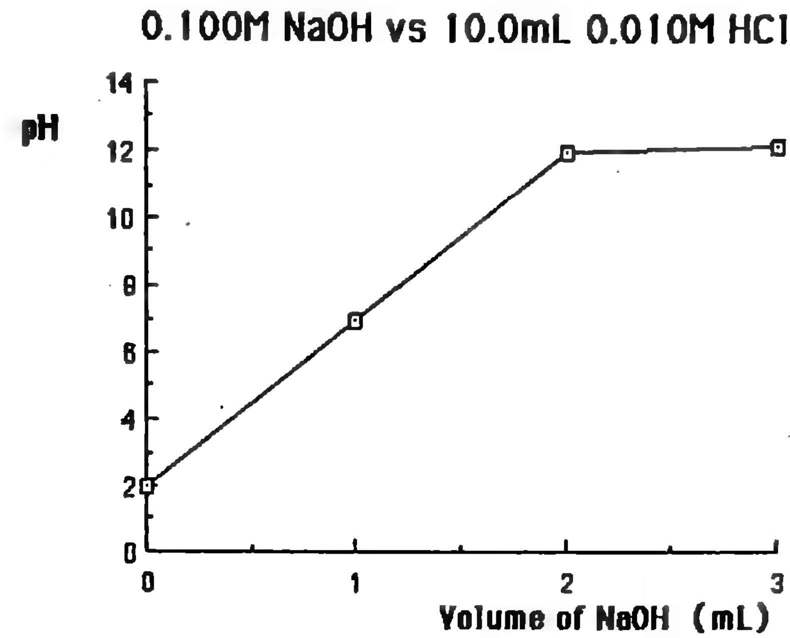


Figure 3.5 Titration curve using four data points.

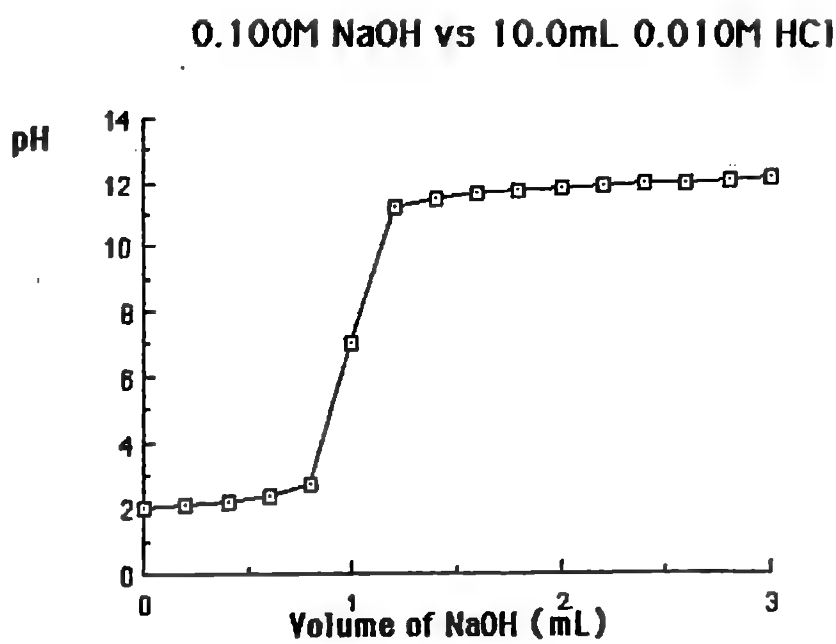


Figure 3.6 Titration curve using a large number of data points.

3. Pascal Code for initializing arrays of data.

The following global variables and types are used to produce a pH curve.

```
CONST
  VOLSCALE = 100;
TYPE
  REALPTS = ARRAY [1..VOLSCALE] OF REAL;
  INTPTS  = ARRAY [0..VOLSCALE] OF INTEGER;
VAR
  VOLPTS: REALPTS;
  PHPTS:  INTPTS;
```

VOLSCALE is a constant defining the length of the horizontal axis in terms of numbers of pixels. The correlation between each pixel in the horizontal direction and the volume of titrant which it represents is calculated in PROCEDURE INITARRAYS and stored in the global array VOLPTS.

```
PROCEDURE INITARRAYS;
VAR  VOLRATIO :REAL;  (* vol increment per pixel *)
       FACTOR,          (* percentage of titration to be plotted *)
       I: INTEGER;

BEGIN
   IF TITRTYPE=DIPROTIC THEN FACTOR:= 3 ELSE FACTOR:= 2;
   VOLRATIO:=(ENDPT1*FACTOR)/VOLSCALE;
   FOR I:= 1 TO VOLSCALE DO VOLPTS[I]:=VOLRATIO*I;
END;
```

For every volume stored in array VOLPTS, the corresponding pH value is calculated. CALCPH is called to calculate the pH value for each volume. The pH values are scaled to integer values representing number of pixels on vertical axis on pH graph. These scaled pH values are stored in array PHPTS.

The ratio between pH values and pixels is determined in PROCEDURE INITCONDITIONS:

```
PROCEDURE INITCONDITIONS(VAR ENDPT1:REAL);
CONST   PHSCALE= 100.00;  (* no. pixels on vertical axis *)
        PHRANGE= 14.0;    (* scale represents a pH range of 0-14*)
.
.
PHRATIO:=(PHSCALE/PHRANGE);  (* pH increment per pixel *)
.
.
```

Integer values for scaled pH are assigned in PROCEDURE CYCLE which is local to PROCEDURE SETUPARRAYS:

```
PROCEDURE CYCLE;
BEGIN
  WHILE ((I<VOLSCALE) AND (NOT KEYIN)) DO
    BEGIN
      I:= I+1;
       IF INFLASK=ACID THEN
          CALCPH(FLASKVOL,VOLPTS[I],HCONC,OHCONC,PH)
          ELSE
          CALCPH(VOLPTS[I],FLASKVOL,HCONC,OHCONC,PH);
       PHPTS[I]:=ROUND(PHRATIO*PH);
    END; (* WHILE *)
END; (*CYCLE*)
```

In the majority of titrations the time required to carry out these calculations is only a few seconds, however in some cases it may take much longer due to the solving of high order polynomial equations in CALCPH. To avoid a long pause while this is being carried out, PROCEDURE SETUPARRAYS calls PROCEDURE CYCLE whenever the program is waiting for user response. Once the user has responded (detected by FUNCTION KEYIN) then PROCEDURE CYCLE is suspended until the program has handled the user input.

4. Pascal Code for plotting smooth pH curve.

Plotting pH curve is carried out by PROCEDURE GRAPH which is called whenever an increment of titrant has been added to the flask. INDEX is a key variable to plotting the pH curve.

INDEX – corresponds to a pixel position along the horizontal axis.
- is an index to the array VOLPTS which contains the unscaled volume of titrant at this pixel position.
- is an index to the array PHPTS which contains the position on pH axis corresponding to titrant volume in VOLPTS.

For example, an INDEX value of 25 refers to the twenty-fifth pixel along the horizontal axis (50% titration for monoprotic acids) which corresponds to a titrant volume found in VOLPTS[25] and to a height on pH axis found in PHPTS[25].

The values used to plot a single point on the graph are:

(i) INDEX (pixels in horizontal direction) and

(ii) PHPTS[INDEX] (pixels in the vertical direction).

Plotting the graph is carried out in the following manner:

1. The pen is moved to the coordinates of the last pixel plotted (OLDX,OLDY)
2. INDEX is incremented and the point INDEX, PHPTS[INDEX] plotted until either
   (i) VOLPTS[INDEX] is greater than the total volume of titrant or
   (ii) INDEX has reached a maximum value.

```
PROCEDURE GRAPH(VAR OLDX,OLDY,INDEX: INTEGER;
               VAR NEXTVOL: REAL; COL: SCREENCOLOR);
VAR EXACTPH,
    X,Y : INTEGER;
BEGIN
  MOVECOL(OLDX,OLDY,COL);        (*move to last point plotted *)
  WHILE (BURVOL>=NEXTVOL) AND (INDEX<VOLSCALE) DO
    BEGIN
      INDEX:=INDEX+1;
      X:=INDEX+XCON; Y:=PHPTS[INDEX]+YCON;
      MOVETO(X,Y);
      NEXTVOL:=VOLPTS[INDEX+1];
      OLDX:=X;
      OLDY:=Y;
    END;
  IF (INDEX<VOLSCALE) THEN
    BEGIN
      EXACTPH:=ROUND(PH*PHRATIO);
      MOVETO(OLDX,EXACTPH+YCON);
      OLDY:=EXACTPH+YCON;
    END;
  PENCOLOR(NONE);
  END; (*GRAPH*)
```

3.3.3 CALCULATION OF PH

The pH of the solution at any point in the titration is carried out by PROCEDURE CALCPH which is found in the Library Unit called Titrlib. A detailed discussion of procedure calcph is presented.

Symbols

The following symbols are used in the discussion regarding calculation of pH:

M = molarity

[ ] = concentration

Ma = initial molarity of acid

Va = volume of acid in solution

Mb = initial molarity of base

Vb = volume of base in solution

Kw = dissociation constant for water

Ka & Kb = dissociation constants of weak acid and base respectively.

K1 & K2 = first and second dissociation constants of diprotic acid

Ca = concentration of acid in solution

Cs = concentration of salt in solution

Cs1 & Cs2 = concentration of mono- and di-substituted salts respectively.

PROCEDURE CALCPH

This procedure is passed the following information in the form of value parameters:

(i) volume of acid in solution
(ii) volume of base in solution
(iii) initial concentration of acid
(iv) initial concentration of base

If a weak acid, weak base or diprotic acid is involved then the

acid dissociation constants are required. This information is obtained from global variables. The information passed to PROCEDURE CALCPH is first manipulated to calculate total volume of solution, moles of acid, moles of base and the molarity of excess acid or base. Boolean variables, EXCESSACID, EXCESSBASE and EQUIVPT are set to indicate the appropriate stage of the titration.

$$\text{total volume} = Va + Vb$$

$$\text{moles acid} = MaVa$$

$$\text{moles base} = MbVb$$

$$\text{excess} = |(MaVa - MbVb)| / (Va + Vb)$$

After these initial calculations which set all the variables local to PROCEDURE CALCPH, a case statement directs the program to one of three different procedures, STRONGCALC, WEAKCALC or DICALC, according to the nature of the titration. TITRTYPE is a global variable which indicates nature of current titration.

```
PROCEDURE CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC:REAL;
                                                        VAR PH:REAL);
CONST
     DIFF=1.0E-6;                 (* minimum  millimoles considered*)
VAR  ACIDMOLS,BASEMOLS,           (* net moles of acid & base in soln*)
     TOTALVOL,                    (* total volume of soln *)
      EXCESS : REAL;              (* conc. of excess acid or base*)
      EXCESSACID,EXCESSBASE,EQUIVPT : BOOLEAN; (* nature of soln *)
       ....
       ....
       ....
BEGIN   (* calcph *)
  TOTALVOL:=ACIDVOL+BASEVOL;
  ACIDMOLS:=ACIDVOL*HCONC;
  BASEMOLS:=BASEVOL*OHCONC;
  EXCESS:=ACIDMOLS-BASEMOLS;
  IF ABS(EXCESS)<DIFF THEN EXCESS:=0.0;
  EXCESSACID:=EXCESS>0;
  EXCESSBASE:=EXCESS<0;
  EQUIVPT:=EXCESS=0;
  EXCESS:= ABS(EXCESS)/TOTALVOL;
  CASE TITRTYPE OF
     STRONG ACID :  STRONGCALC;
     WEAKACID,
     WEAKBASE    :  WEAKCALC;
     DIPROTIC    :  DICALC;
     END;  (* case *)
END;  (* calcph *)
```

PROCEDURE STRONGCALC is responsible for determining the pH of solutions formed at any point of a titration between a strong acid and strong base. For a solution with excess strong acid, hydrogen ions result from the dissociation or ionization of the excess acid and also from the ionization of water.

$$[H^+]total = [H^+]acid + [H^+]water$$

The concentration of hydrogen ions resulting from the acid is the same as the concentration of the strong acid. The concentration of hydrogen ions due to the ionization of water is determined by solving the following quadratic equation:

$$[H^+]^2 water + [H^+]acid\ [H^+]water - Kw = 0$$

Except for very dilute solutions the contribution of hydrogen ions from the dissociation of water is insignificant compared with the contribution from the strong acid. PROCEDURE STRONGCALC solves the quadratic equation whenever the concentration of acid is less than 1.0 x 10^{-6} molar.

Similar calculations are involved to determine the concentration of hydroxide ions in a solution with excess base:

$$[OH^-]total = [OH^-]base + [OH^-]water$$

and

$$[OH^-]^2 water + [OH^-]base \ [OH^-]water - Kw = 0$$

```
PROCEDURE STRONGCALC;
CONST
 DILUTE=1.0E-6;

   PROCEDURE CORRECT(VAR VALUE:REAL);
   (*consider contribution of ions from the dissociation of water *)
   VAR DISCR:REAL;
   BEGIN
       DISCRIM:=SQRT((VALUE*VALUE) + 4*KW);
       VALUE:=(VALUE + DISCRIM) / 2;
   END;

BEGIN
  IF EXCESS<DILUTE THEN CORRECT(EXCESS)
  PH:=-LOG(EXCESS);
  IF EXCESSBASE THEN PH:=14-PH
END; (*STRONGCALC*)
```

PROCEDURE WEAKCALC calculates the pH of solutions formed at any point of a titration between a weak acid and a strong base or a strong acid and a weak base. The titrant may be either the acidic or basic solution.

At various stages in the titration between a weak acid and a strong base, the solution will contain one of the following:

(i) a weak acid;

(ii) a salt of weak acid;

(iii) a weak acid plus its salt;

(iv) a strong base plus salt of weak acid;

(v) a strong base;

Similarly for titrations between a weak base and a strong acid the resulting solutions will contain one of the following:

(i) a weak base;

(ii) a salt of a weak base;

(iii) a weak base plus its salt;

(iv) a strong acid plus salt of weak base;

(v) a strong acid;

PROCEDURE WEAKCALC classifies the above solutions into two groups:

1. If there is excess strong acid in the solution, then the hydrogen ions result from three sources :-

   (i) ionization of the strong acid,

   (ii) hydrolysis of the salt of a weak base and

   (iii) self-ionization of water.

$$[H^+]\text{total} = [H^+]\text{acid} + [H^+]\text{salt} + [H^+]\text{water}$$

   Similarly for a solution with excess strong base:

$$[OH^-]\text{total} = [OH^-]\text{base} + [OH^-]\text{salt} + [OH^-]\text{water}$$

   Except for extremely dilute solutions of acid or base and salt, the contribution of hydrogen ions from the self-

ionization of water is insignificant. The acidity is first evaluated considering only the concentration of salt and acid or base:

$$[H^+] = [\text{strong acid}] + \text{SQRT}(Kw*Cs/Ka)$$

or $$[OH^-] = [\text{strong base}] + \text{SQRT}(Kw*Cs/Kb)$$

Only if the evaluated concentration of hydrogen ions is within the range of 10^{-6} to 10^{-8} molar is the self-ionization of water considered. PROCEDURE WEAKCALC calls PROCEDURE HYDROLYSIS to solve the above equations.

2. In all other cases the solution contains a weak acid or base and/or the salt of a weak acid or base. PROCEDURE WEAKCALC calculates the concentration of acid and salt in the solution and approximates concentration of hydrogen or hydroxide ions. The following equations are used to estimate the acidity:

Weak acid	approximate [H+] =	SQRT(Ka*Ca)
Salt of a weak acid	approximate [H+] =	SQRT(Ka*Kw/Cs)
Weak acid + its salt	approximate [H+] =	Ka*Ca/Cs
Weak base	approximate[OH-] =	SQRT(Kb*Cb)
Salt of a weak acid	approximate[OH-] =	SQRT(Kb*Kw/Cs)
Weak acid + its salt	approximate[OH-] =	Kb*Cb/Cs

The exact concentration of hydrogen ions or hydroxide ions is determined by passing the approximate value to

PROCEDURE SOLVEQN which solves the following cubic equation:

$$[H^+]^3 + (Cs + Ka)[H^+]^2 - (Ka*Ca + Kw)[H^+] - Ka*Kw = 0$$

or

$$[OH^-]^3 + (Cs + Kb)[OH^-]^2 - (Ka*Cb + Kw)[OH^-] - Kb*Kw = 0$$

(PROCEDURE SOLVEQN, discussed on page 93, uses the Newton-Raphson method to solve the cubic equation, with the initial guess being the approximate concentration of hydrogen ions or hydroxide ions.)

```
PROCEDURE WEAKCALC;
VAR
 SALT1CONC ,SALT2CONC, APPROXH : REAL;
 WEAKEXCESS : BOOLEAN;

    PROCEDURE HYDROLYSIS(SALT; VAR H,PH:REAL);
    CONST DILUTE=1.0E-6
    VAR SALTHYD : REAL;
    BEGIN (*HYDROLYSIS*)
      SALTHYD:=SQRT(KW*SALT/K1);  (*hydrolysis of salt*)
      H:=H + SALTHYD;
      IF H<DILUTE THEN H:=(H + SQRT((H*H)+4*Kw))/2;  (*hydrol of*)
      PH:=-LOG(H);                                    (* water*)
    END; (*HYDROLYSIS*)

BEGIN   (*WEAKCALC*)
  SALT2CONC:=0.0;
  IF EXCESSACID THEN SALT1CONC:=BASEMOLS/TOTALVOL
      ELSE  SALT1CONC:= ACIDMOLS/TOTALVOL;
  WEAKEXCESS:=((TITRTYPE=WEAKACID) AND (EXCESSACID)) OR
              ((TITRTYPE=WEAKBASE) AND (EXCESSBASE));

  IF EQUIVPT  THEN  APPROXH:=SQRT(K1*KW/SALT1CONC)
     ELSE   IF  SALT=0.0 THEN  APPROXH:=SQRT(K1*EXCESS)
       ELSE  APPROXH:=K1*EXCESS/SALT1CONC;

  IF WEAKEXCESS OR EQUIVPT THEN
     SOLVEQN(EXCESS,SALT1CONC,SALT2CONC,APPROXH,PH)
         ELSE HYDROLYSIS(SALT1CONC,EXCESS,PH);
  IF EXCESSBASE OR (EQUIVPT AND TITRTYPE=WEAKBASE) THEN
    PH:=14 - PH;
END;  (*WEAKCALC*)
```

PROCEDURE DICALC calculates the pH of any solution formed during the titration of a diprotic acid with a strong base. The titrand is restricted to being the diprotic acid.

At various stages in the titration the solution will contain one of the following:

(i) diprotic acid;
(ii) diprotic acid plus its monosubstituted salt;
(iii) monosubstituted salt of diprotic acid;
(iv) mono and di-substituted salts of diprotic acid;
(v) disubstituted salt of diprotic acid;
(vi) strong base plus disubstituted salt;

These solutions are classified into two groups:

1. Where the solution contains excess base. In this case the hydroxide ions arise from dissociation of the base, hydrolysis of the salt and hydrolysis of water. Due to the presence of the disubstituted salt, the solution will always be reasonably basic, and the contribution of hydroxide ions from water is neglected:

$$[OH^-]total = [strong\ base] + SQRT(Kw*Cs2/K2)$$

PROCEDURE DICALC calls PROCEDURE HYDROLYSIS to calculate the concentration of hydroxide ions in solution according to the above equation.

2. In all other cases the solution contains a diprotic acid, a monosubstituted salt, a disubstituted salt or some combination of the three. PROCEDURE DICALC approximates the acidity of the solution according to the following equations:

Diprotic acid	approx [H+]	= SQRT(K1*Ca)
Diprotic acid + mono-salt	approx[H+]	= K1*Ca/Cs1
Monosubstituted salt	approx [H+]	= SQRT(K1*K2)
Mono- and di-salts	approx [H+]	= K2*Cs1/Cs2
Disubstituted salt	approx [H+]	= SQRT(Kw*K2/Cs2)

This approximate acidity is passed to PROCEDURE SOLVEQN, which uses this value as the initial guess in solving the following quartic equation by the Newton-Raphson method:

$$[H^+]^4 + (K1+Cs1+ 2*Cs2)[H^+]^3 + (K1*K2-K1*Ca+K1*Cs2 - Kw)[H^+]^2 - (K1*Kw + 2*K1*K2*Ca + K1*K2*Cs1)[H^+] - K1*K2*Kw = 0$$

```
PROCEDURE DICALC;
VAR  ACIDCONC,BASECONC,SALT1CONC,SALT2CONC,
      APPROXH :REAL;

    PROCEDURE HYDROLYSIS(SALT:REAL; VAR OH,PH:REAL);
    VAR SALTHYD:REAL;
    BEGIN
      SALTHYD:=SQRT(KW*SALT/K2);
       OH:= OH + SALTHYD;
       PH:=14+LOG(OH);
    END;

BEGIN (*DICALC*)
   IF BASEMOLS>2*ACIDMOLS THEN (*past 2nd endpt-excess strong base*)
      BEGIN
        SALT2CONC:= ACIDMOLS/TOTALVOL;
         BASECONC:=(BASEMOLS-(2*ACIDMOLS))/TOTALVOL;
         HYDROLYSIS(SALT2CONC,BASECONC,PH);
       END
       ELSE
      BEGIN
         IF EXCESSBASE THEN (*between 1st and 2nd end pt - two salts*)
         BEGIN
            ACIDCONC:=0.0;
            SALT1CONC:=(2*ACIDMOLS-BASEMOLS)/TOTALVOL;
            SALT2CONC:=EXCESS;
            IF SALT1CONC:=0 THEN APPROXH:=SQRT(KW*K2/SALT2CONC)
                 ELSE APPROXH:=K2*SALT1CONC/SALT2CONC;
         END
         ELSE        (* from start to 1st end pt - acid and salt*)
           BEGIN
             ACIDCONC:=EXCESS;
             SALT1CONC:=BASEMOLS/TOTALVOL;
             SALT2CONC:=0.0;
             IF SALT1CONC=0 THEN APPROXH:=SQRT(K1*ACIDCONC)
                 ELSE IF ACIDCONC:=0 THEN APPROXH:= SQRT(K1*K2)
                      ELSE APPROXH:=K1*ACIDCONC/SALT1CONC;
           END;
     SOLVEQN(ACIDCONC,SALT1CONC,SALT2CONC,APPROXH,PH);
   END;
END; (*DICALC*)
```

PROCEDURE SOLVEQN defines the coefficients of the polynomial (a,b,c,d, and e) in the form of equations relating to a fourth order polynomial appropriate for a mixture of diprotic acid and its two salts. The cubic equation required for solutions involving monoprotic acids or bases and related salts, is a special case, which is derived from this general fourth order polynomial. The general form:

$$[H^+]^n + (K1 + Cs1 + 2*Cs2)[H^+]^{n-1} + (K1*K2 - K1*Ca + K1*Cs2 - Kw)[H^+]^{n-2} - (K1*Kw + 2*K1*K2*Ca + K1*K2*Cs1)[H^+]^{n-3} - K1*K2*Kw = 0$$

This equation yields the following quartic equation when n=4:

$$[H^+]^4 + (K1 + Cs1 + 2*Cs2)[H^+]^3 + (K1*K2 - K1*Ca + K1*Cs2 - Kw)[H^+]^2 - (K1*Kw + 2*K1*K2*Ca + K1*K2*Cs1)[H^+] - K1*K2*Kw = 0$$

and yields the following cubic equation when n=3 and K2=0:

$$[H^+]^3 + (Cs + Ka)[H^+]^2 - (Ka*Ca + Kw)[H^+] - Ka*Kw = 0$$

Underflow errors

If the hydrogen ion concentration is less than 10^{-9} molar, then calculation of $[H^+]^4$ will cause an exponential underflow error in the Apple 2 computer. This problem was avoided by scaling $[H^+]$ by a factor of 1×10^5.

Quartic equation: $a[H^+]^4 + b[H^+]^3 + c[H^+]^2 + d[H^+] + e = 0$

becomes: $aX^4 + b*10^5*X^3 + c*10^{10}*X^2 + d*10^{15}X + e*10^{20} = 0$

where $X = [H^+] * 10^5$

The coefficents for the polynomial, and the initial estimate of hydrogen ion concentration are passed from PROCEDURE SOLVEQN to PROCEDURE NEWTON to calculate the hydrogen

ion concentration.

```
PROCEDURE SOLVEQN(ACID,SALT1,SALT2,GUESS:REAL;VAR PH:REAL);
CONST POWER= 1E5;
VAR  A : ARRAY [1..5] OF REAL;
     NUM,SCALE : INTEGER;
BEGIN
   A[1]:=1.0
   A[2]:= K1 + SALT1 + 2*SALT2;
   A[3]:= K1*K2 - K1*ACID + K1*SALT2 - KW;
   A[4]:= -(K1*KW  + 2*K1*K2*ACID + K1*K2*SALT1)
   A[5]:= -(K1*K2*KW);
  FOR NUM:=2 TO 5 DO
     FOR SCALE:=1 TO NUM-1 DO  A[NUM]:=A[NUM]*POWER;
   GUESS:=GUESS*POWER;
   IF TITRTYPE=DIPROTIC THEN
     NEWTON(A[1],A[2],A[3],A[4],A[5],GUESS,H)
       ELSE NEWTON(0,A[1],A[2],A[3],A[4],GUESS,H);
 H:=(H/POWER);
END; (*SOLVEQN*)
```

PROCEDURE NEWTON

The Newton-Raphson method is one of the most widely used iterative methods for evaluating roots of equations. Given the function f(x)=0 and the initial approximation to the root x_0 the Newton-Raphson method is

$$x_{i+1} = x_i - [f(x_i) / f'(x_i)]$$

$f(x_i)$ and $f'(x_i)$ are the function and its first derivative evaluated at $x=x_i$. This method requires an initial estimation of the root, x_0, which should be as close to the desired root as possible. The estimation of hydrogen ions is determined in PROCEDURES WEAKCALC and DICALC. This value is ultimately passed onto PROCEDURE NEWTON.

The convergence criterion is that

$$| (x_i - x_{i+1}) / x_{i+1} | < \text{SPECIFIED TOLERANCE}$$

PROCEDURE NEWTON applies the Newton-Raphson method to any polynomial of fourth order or lower:

$$[H^+]_{i+1} = [H^+]_i - [f([H^+]_i) / f'([H^+]_i)]$$

where $f([H^+]) = a[H^+]^4 + b[H^+]^3 + c[H^+]^2 + d[H^+] + e$

and $f'([H^+]) = 4a[H^+]^3 + 3b[H^+]^2 + 2c[H^+] + d$

The iteration cycle ceases when the convergence criterion is met, or a maximum number of iterations has been performed. If the method does not converge, or if a negative root has been found, then the initial guess is altered and the iteration cycle restarted.

```
PROCEDURE NEWTON(A,B,C,D,E,APPROX:REAL; VAR PH:REAL);
CONST CRITERIA:=0.001;
VAR COUNT: INTEGER;
     NEWTONX,ERROR, GUESS : REAL;
     SOLN: BOOLEAN;

        FUNCTION EQUATION(X:REAL):REAL;
        BEGIN
           EQUATION:=E+X*(D + X*(C+X*(B+A*X)));
        END;

        FUNCTION DERIV(X:REAL):REAL;
        BEGIN
           DERIVE:= D+X*(2*C + X(3*B + 4*A*X));
        END;

BEGIN (*NEWTON*)
COUNT:=0;
GUESS:=APPROX;
SOLN:=FALSE;
REPEAT
   COUNT:=COUNT+1;
    NEWTONX:=APPROX-(EQUATION(APPROX)/DERIV(APPROX));
   IF ABS((NEWTONX-APPROX)/NEWTONX) < CRITERIA THEN SOLN:=TRUE
      ELSE APPROX:=NEWTONX;
 UNTIL ((COUNT>20) OR (SOLN));
 IF (NEWTONX<0) OR (NOT SOLN) THEN
  BEGIN
      APPROX:=GUESS*10;
      NEWTON(A,B,C,D,E,APPROX);
    END    ELSE   PH:=-LOG(NEWTONX);
 END; (*NEWTON*)
```

CHAPTER 4. SALT TITRATION PACKAGE

4.1 DESCRIPTION OF SALT TITRATION PACKAGE

4.1.1 User interface for package

4.1.2 Titration of Salts Program

4.1.3 Titration of a Mixture of Sodium Carbonate & Sodium Bicarbonate Program

4.1.4 Assignment for Salts Program

4.1.5 Assignment for Mixtures Program

4.2 OBJECTIVES OF THE SALT TITRATION PACKAGE

4.2.1 Worksheets

4.3 PASCAL CODE FOR SALT TITRATION PACKAGE

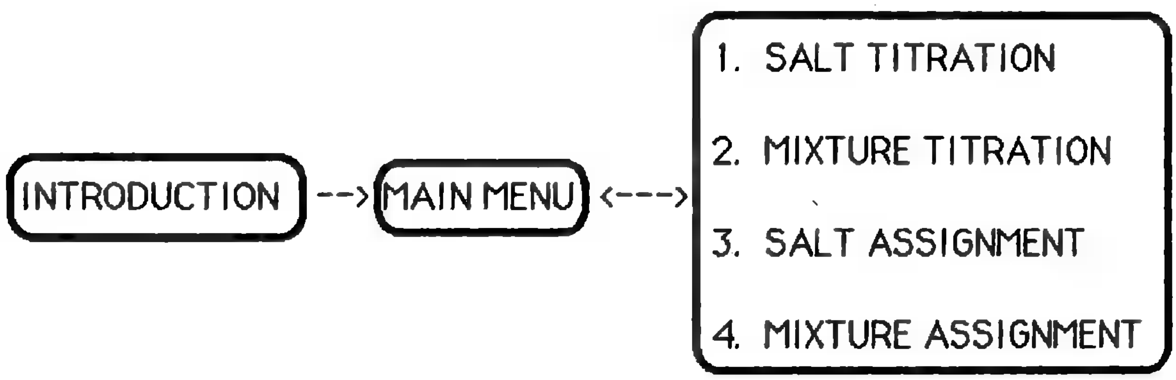
4.1 DESCRIPTION OF SALT TITRATION PACKAGE

This package is an extension of the Acid/Base Titration package. The SALT TITRATION PACKAGE consists of four programs each of which involves the titration of acidic and basic salts.

4.1.1 USER INTERFACE FOR SALT TITRATION PACKAGE

The SALT TITRATION PACKAGE appears to the user in three main sections:

- an introduction section
- a main menu
- four titration programs. Each of these four programs links back to the main menu for the package.



The SALT TITRATION PACKAGE containing the above programs is located on one disk. This disk must always be resident in the disk drive as it is necessary to access the disk each time the program moves from one program to another.

Introduction to package

The introduction consists of four pages on the graphics screen. These introduction pages are only encountered when the disk is first booted. Except for the title page, the introduction section for the SALT TITRATION PACKAGE is the same as the introduction section for the ACID/BASE TITRATION PACKAGE (Section 3.1.1).

Main menu:

This package is menu driven with the following main menu:

Titration of Salts	....(1)
Titration of Mixture of Sodium Carbonate & Sodium Bicarbonate	....(2)
Assignment for Salts	....(3)
Assignment for Mixture	....(4)
Quit	....(Q)

Four salt titration programs

The four programs are discussed in detail in the following sections. Entering "Q" from within any of the above programs will eventually bring the user back to the main menu for the package.

4.1.2 TITRATION OF SALTS PROGRAM

This program allows the user to carry out a simulated titration between an acidic or basic salt solution, and a strong acid or base. A variety of indicators is available for each titration. Throughout the titration the pH of the solution is displayed and a graph of pH against volume of titrant is also produced.

HOW TO SPECIFY A PARTICULAR TITRATION

Option 1. Type of salt

The main menu for this program offers the following categories of salt:

1. Salt of a weak acid/strong base.
2. Salt of a weak base/strong acid.
3. Salt of a diprotic acid/strong base.

Option 2. Nature of salt

The variety of salt solutions presented depends on the type of salt selected in option 1. One of the following menus will be displayed for selection of salt:

Weak acid/strong base salts (if "1" was chosen in option 1):

1. sodium cyanide
2. sodium acetate
3. input pKa of acid from which salt is derived

Weak base/strong acid salts (if "2" was chosen in option 1):

1. ammonium chloride
2. input pKb of base from which salt is derived

Diprotic acid/strong base salts (if "3" was chosen in option 1):

1. sodium carbonate

2. potassium phthalate
3. input pK1 & pK2 of acid from which salt is derived

Option 3. Titrant and Titrand

In this program the salt solution is always the titrand. The user must specify the volume of the salt solution, in the range of 10.00mL to 50.00mL. If an acidic salt is chosen then the titrant will be sodium hydroxide solution otherwise the titrant is hydrochloric acid. The user must determine the concentration of the salt solution and the titrant.

Option 4. Indicators

The following indicators are available:

1. methyl orange
2. methyl red
3. litmus
4. bromothymol blue
5. phenolphthalein
6. thymolphthalein
7. ideal

After the above options have been determined the simulated titration is ready to commence.

SIMULATED TITRATION FOR TITRATION OF SALTS

1. Screen layout

Throughout the simulated titration the following information is displayed on the screen:

(i) concentration of salt
(ii) concentration of titrant
(iii) volume of salt in flask

(iv) volume of titrant in flask

(v) pH of solution in flask

(vi) graph of pH vs. volume of titrant showing colour range for selected indicator.

Repeat options

The titration may be continued (by repeatedly pressing SPACE BAR) until either "Q" is pressed, indicating that the user wishes to quit from the titration, or until the flask becomes full. When the flask is full the user is forced to quit from the titration. After quitting from the titration the user is given the following options:

(i) Repeat the previous titration with the same indicator.

(ii) Repeat the previous titration with a different indicator.

(iii) Carry out a different titration.

(iv) Quit from the *Salt* program, which takes the user back to the main menu for the package.

4.1.3 TITRATION OF A MIXTURE OF SODIUM CARBONATE & SODIUM BICARBONATE PROGRAM

In this titration a mixture of sodium carbonate and sodium bicarbonate is titrated with standard solution of hydrochloric acid.

HOW TO SPECIFY A MIXTURE OF CARBONATE & BICARBONATE

Option 1. Carbonate and bicarbonate

The user must specify the concentration of both the carbonate and bicarbonate in the salt solution. The maximum concentration of either salt is 1.000 molar. It is acceptable to specify the concentration of one of the salts as zero, resulting in a solution containing only one salt.

Option 2. Concentration of Titrant

The titrant for this titration is hydrochloric acid. The user must specify the concentration of hydrochloric acid, in the range of 0.001 molar to 1.000 molar.

Option 3. Indicators

The following indicators are available:

1. methyl orange
2. methyl red
3. bromothymol blue
4. phenolphthalein
5. ideal (1st equiv. pt)
6. ideal (2nd equiv. pt)

SIMULATED TITRATION FOR TITRATION OF MIXTURE

Throughout the simulated titration the following information is displayed on the screen:

(i) concentration of sodium carbonate & sodium bicarbonate

(ii) concentration of hydrochloric acid

(iii) volume of carbonate/bicarbonate mixture in flask

(iv) volume of hydrochloric acid in flask

(v) pH of solution in flask

(vi) graph of pH vs. volume of titrant showing colour range for selected indicator.

Repeat options

After quitting from the titration the user is given the following options:

(i) Repeat the previous titration with the same indicator.

(ii) Repeat the previous titration with a different indicator.

(iii) Select a different titration.

(iv) Quit from this program, back to main menu.

4.1.4 ASSIGNMENT FOR SALTS PROGRAM

This program provides acidic and basic salt solutions as assignment solutions of unknown concentration. The user must carry out a series of titrations in order to determine the concentration of each assignment solution. An ideal indicator is used to determine the end point of the titration. The program does not inform the user of the concentration of the solution involved in each assignment, and a written report on the titration data and calculations necessary to determine this concentration may be submitted to the teacher for evaluation.

Option 1. Assignment number

The user is prompted to enter an assignment number between 1 and 99. This number is used by the program to determine the nature and concentration of the assignment solution. The assignment solutions are sodium acetate and sodium carbonate. The concentration of the solution will be between 0.001 molar and 1.000 molar.

Option 2. Concentration of standard solution

The nature of the standard solution is determined by the program however the user must enter the concentration of the standard solution.

Option 3. Volume of salt solution

The program determines that the salt solution will always be the titrand, however the user must determine the volume of the salt solution.

Repeat options

After quitting from the titration the user is given the following options:

(i) Repeat the previous titration with the same conditions.

(ii) Repeat the previous titration but alter the conditions.

(iii) Get the calculator.

(iv) Start a new assignment.

(v) Quit this program, back to the main menu.

4.1.5 ASSIGNMENT FOR MIXTURE PROGRAM

The assignment solution in this program is a mixture of sodium carbonate and sodium bicarbonate. Hydrochloric acid is the standard solution which will be the titrant. A written report on the estimated concentration of assignment solution may be submitted to the teacher for assessment - the program does not inform the student of this concentration.

Option 1. Assignment Number

The user is prompted to enter a number between 1 and 99. The assignment number corresponds to particular concentrations of carbonate and bicarbonate.

Option 2. Concentration of standard solution

The user must specify the concentration of hydrochloric acid.

Option 3. Indicator

The program offers a choice of two ideal indicators. One indicator changes colour at the equivalence point for the carbonate and the second changes colour at the equivalence point for the bicarbonate.

Option 4. Volume of salt solution

The user must determine the volume of the salt mixture solution.

Repeat options

After quitting from the titration the user is given the following options:

(i) Repeat the previous titration with the same conditions.
(ii) Repeat the previous titration but alter the conditions.
(iii) Get the calculator.

(iv) Start a new assignment.

(v) Quit this program, back to the main menu.

4.2 OBJECTIVES OF THE SALT TITRATION PACKAGE

In order to solve problems relating to the titration of salts, the student must fully understand the concepts involved in a simple acid/base titration. Therefore, it is preferrable that students be exposed to the ACID/BASE TITRATION PACKAGE before commencing the SALT TITRATION PACKAGE.

Major objectives of the SALT TITRATION PACKAGE:

1. Firstly this package was designed to enable the student to develop problem solving skills. Two of the programs are assignment programs, containing in total two hundred sample solutions for which simulated titrations can be performed to find the concentration.
2. Secondly the package enables the student to investigate a number of titration variables. This is achieved by carrying out a series of titrations in which one variable at a time is studied.

1. Titration of Salts

This program introduces the fundamental concepts of a titration involving an acidic or basic salt solution. A major objective of this program is to encourage the student to investigate the relationship between a weak acid (or base) and its salt. The simulation allows the user to select the acidic strength of the salt as a function of the strength of the weak acid (or base) from which it is derived.

Worksheets for this program require titration of a number of salt solutions of varying acidity, and also titrations of the

corresponding weak acids (and bases) from which these salts are derived.

2. Titration of a mixture of sodium carbonate and bicarbonate.

Sodium carbonate is a common primary standard and therefore, sodium carbonate titrations are extremely important in chemical analysis. Major objectives of this program are to:

- illustrate the features of the pH curve obtained from a titration involving sodium carbonate.
- allow the student to investigate the relationship between the two end points of a sodium carbonate titration.
- allow the student to study the relationship between the two end points when the solution contains a mixture of bicarbonate and sodium bicarbonate.
- illustrate the most appropriate indicator for each end point.

3. Assignment for salts and assignment for mixture

The assignment programs provide numerous problem solving tasks and also reinforce concepts introduced in earlier programs. Through repeated titrations the student is able to develop (simulated) techniques required to determine the concentration of relatively complex solutions.

The assignment programs are designed for individual or small group tutorials and may also be used for student assessment. The programs provide excellent pre- and post-laboratory exercises.

4.2.1 WORKSHEETS

A series of worksheets has been designed to complement the SALT TITRATION PACKAGE. In order to complete the worksheets, the student must carry out a number of simulated titrations involving acidic or basic salt solutions. Utilization of the worksheets encourages the student to experiment with titration variables and thereby derive the relationship between these factors. The worksheets are listed in Appendix A.

4.3 PASCAL CODE FOR SALT TITRATION PACKAGE

Two customised intrinsic library units, Useful Unit and Saltlib Unit, are incorporated into System.Library for use by the SALT TITRATION PACKAGE. The Useful Unit is discussed in Section 2.2.3 and the Saltlib is very briefly discussed in Section 4.3.1. In addition to the system files, the following object code files are included in this package:

System.startup	(code for introduction section)
Menu.code	(code for main menu)
Salttitrate.code	(code for salt program)
Saltmixture.code	(code for mixture of carbonate/ bicarbonate program)
Saltassign.code	(code for salt assignment program)
Mixassign.code	(code for mixture assignment program)

The source code for all programs in the SALT TITRATION PACKAGE are listed in Appendix D.

4.3.1 SALT TITRATION LIBRARY UNIT

Saltlib is a library unit used by all programs in the SALT TITRATION PACKAGE. The saltlib unit is a modification of the Titrlib unit. A number of routines are indentical in both units. The names of several procedures have been altered to ensure that code is easily read. For example, the Titrlib Unit contains procedures ACIDMOLARITY and BASEMOLARITY to display, on the graphics screen, the molarity of the two solutions involved in titration. Procedures SALTMOLARITY and TITRMOLARITY in the Saltlib Unit perform exactly the same task but are now appropriately named for the solutions involved in titrations of salts.

A discussion of the routines in the Saltlib Unit is not necessary due to the similarity with routines in the Titrlib Unit which are fully described in Section 3.3.1.

CHAPTER 5. MICRO/MACRO CHEM DEMONSTRATION PACKAGE

5.1 DESCRIPTION OF MICRO/MACRO CHEM DEMONSTRATION PACKAGE

- 5.1.1 User interface for micro/macro package
- 5.1.2 Acid + Active Metal Program
- 5.1.3 Water + Very Active Metal Program
- 5.1.4 Acid + Carbonate Program
- 5.1.5 Litmus Program

5.2 OBJECTIVES OF THE MICRO/MACRO CHEM DEMONSTRATION PACKAGE

- 5.2.1 Macroscopic demonstrations
- 5.2.2 Microscopic demonstrations
- 5.2.3 Worksheets

5.3 PASCAL CODE FOR MICRO/MACRO CHEM DEMONSTRATION PACKAGE

- 5.3.1 Macroscopic demonstrations
- 5.3.2 Microscopic demonstrations

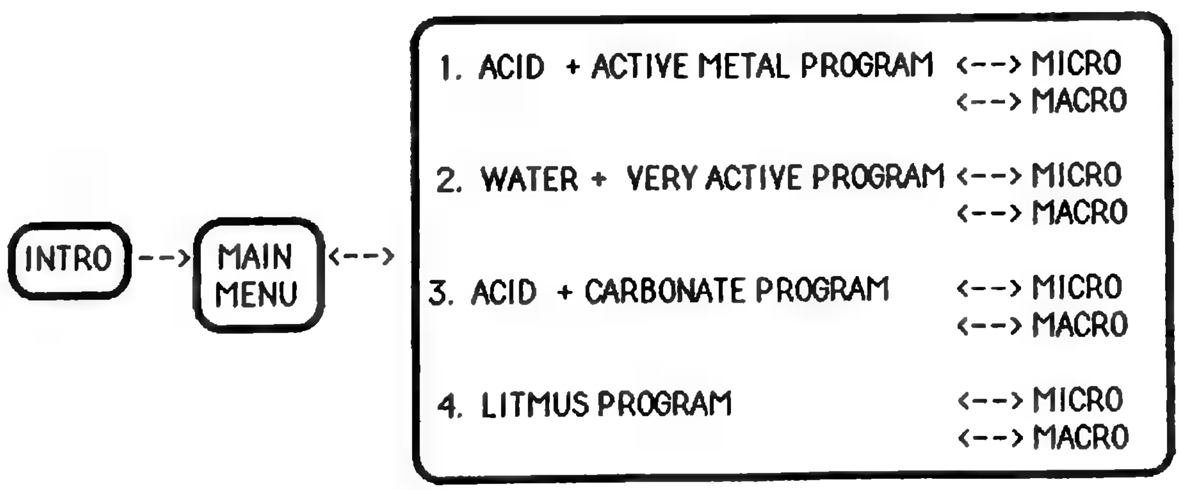
5.1 DESCRIPTION OF MICRO/MACRO CHEM DEMONSTRATION PACKAGE

Each program in this package describes a particular chemical reaction. Within each program there are two demonstrations relating to the same chemical reaction. One demonstration describes the chemical reaction on a macroscopic scale and the other on a microscopic scale. The macroscopic demonstration is simply an animation of the chemical reaction as may be observed in the laboratory. The microscopic demonstration takes this same chemical reaction and displays how the atoms, molecules and ions are reacting. Greater emphasis is placed on presentation of the microscopic demonstrations, as these displays provide a unique teaching aid for the illustration of chemical reactions.

5.1.1 USER INTERFACE FOR MICRO/MACRO PACKAGE

The MICRO/MACRO CHEM DEMONSTRATION PACKAGE appears to the user in three main sections:

- an introduction section
- a main menu
- four programs each of which links back to the main menu.



Introduction

The introduction consists of five pages on the graphics screen. These introductory pages are only encountered when the disk is first booted.

Page 1.	Title page.
Page 2.	Asks whether a colour monitor is being used.
Page 3.	Informs user that it is necessary to press space bar key to progress through the series of programs.
Page 4.	Explanation that each program consists of a microscopic and macroscopic perspective of the one chemical reaction.
Page 5.	Informs user that entering "Q" is required to exit from programs.

An important feature of the microscopic demonstrations, is that by pressing the space bar, the animation on the screen is frozen. This allows close examination of the critical parts of the reaction.

5.1.2 ACID + ACTIVE METAL PROGRAM

This program describes the reaction between a dilute acid and an active metal such as magnesium or iron. The metal dissolves and hydrogen gas is evolved.

Macroscopic Demonstration.

A piece of metal is added to a test tube of hydrochloric acid. The metal dissolves and a gas is evolved. (Figure 5.1) The gas is tested with a flame, causing the gas to explode. This reaction is carried out twice - once with magnesium and once with nickel. The only difference between these two reactions is that the magnesium dissolves more rapidly than the nickel.

A general word equation is presented as a summary of the reactions:

ACID + ACTIVE METAL ---> HYDROGEN + A SALT

This is followed by a series of equations for specific acids and metals.

e.g. SULFURIC ACID + ALUMINIUM ---> HYDROGEN + ALUMINIUM SULFATE

Microscopic Demonstration

Introduction: The demonstration commences by defining the shapes which will be used to represent the water molecules and hydronium ions in the demonstration. (Figure 5.2)

Metallic structure: The structure of magnesium atoms is illustrated followed by a brief description of crystal structure and metallic bonding.

Reaction between hydronium ions and an active metal:

The main part of this demonstration is an animation between an acid and an active metal. To simplify the reaction at this stage

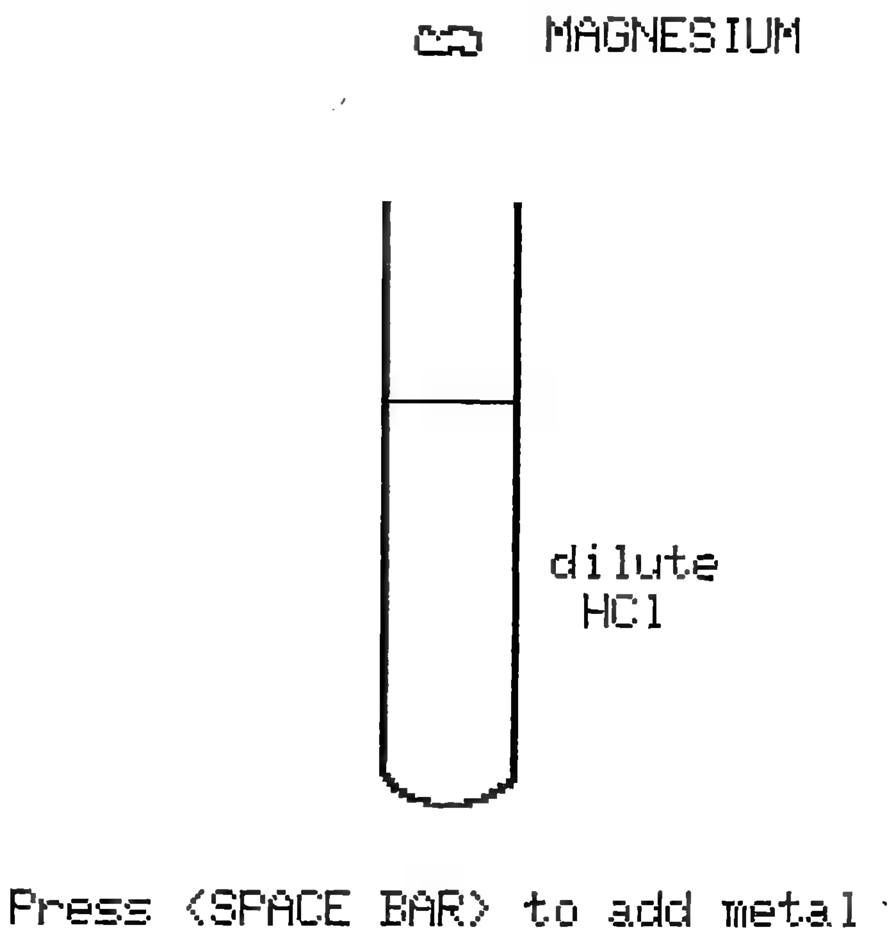


Figure 5.1 Metal Program: a piece of magnesium is dropped into a test tube of hydrochloric acid.

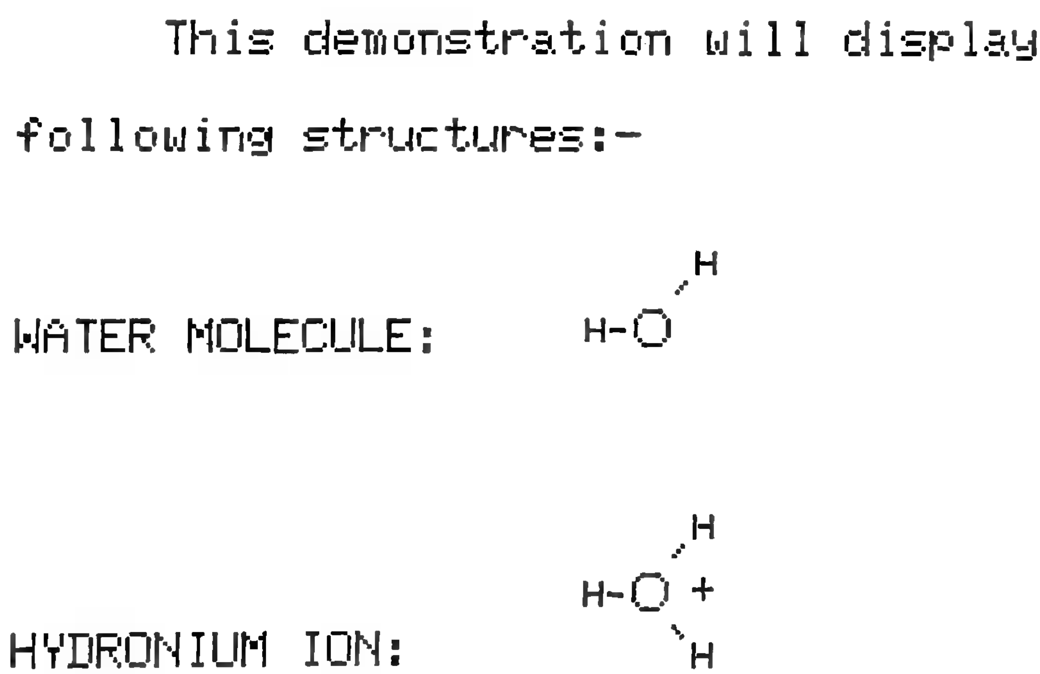


Figure 5.2 Metal Program: definition of shapes of water molecules and hydronium ions displayed in microscopic demonstration.

the spectator ions are not displayed.

Hydronium ions strike the atoms on the surface of the metal. (Figure 5.3) As a result of this collision the valence electrons of the metal atoms are transferred to the hydronium ions. The hydronium ions are converted into hydrogen and water and the metal atoms are converted into metal ions. (Figure 5.4)

Two metals are used in this demonstration. Firstly magnesium and then nickel. In the reaction with magnesium a reaction results from every collision between a hydronium ion and a metal atom, whereas with nickel, not all collisions result in a reaction. This is an attempt to indicate the nickel is a less reactive metal than magnesium.

<u>Summary of reaction:</u> The net reaction of the metal atoms and the hydronium ions are graphically displayed.

The reaction between each metal and hydronium ions is expanded to display spectator ions.

<u>Solvation of ions:</u> The simplification regarding solvation of ions is discussed, and hydrated metal ions are displayed.

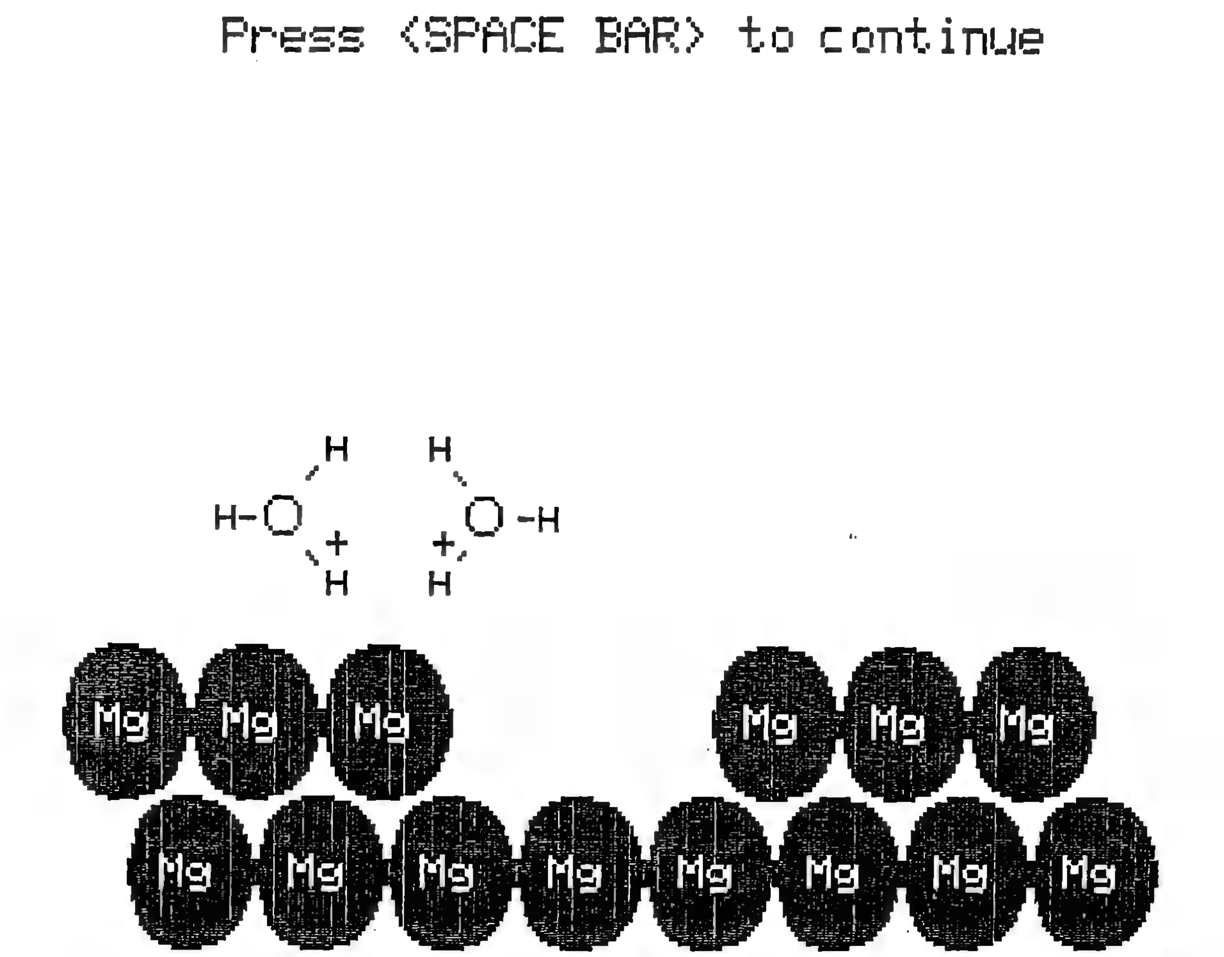


Figure 5.3 Metal Program: hydronium ions react with magnesium metal.

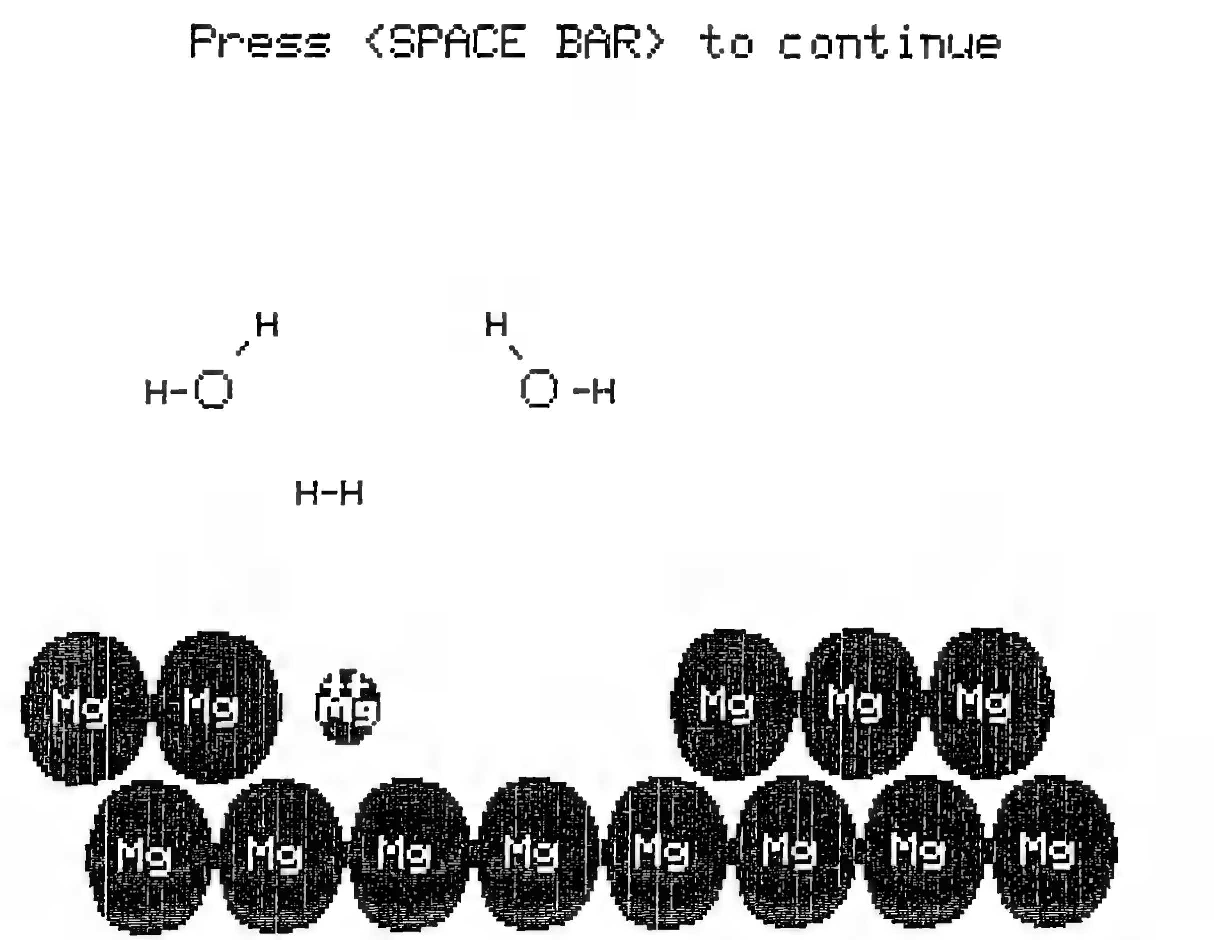


Figure 5.4 Metal Program: hydrogen and water are formed as magnesium loses electrons to hydronium ions.

5.1.3 WATER + VERY ACTIVE METAL PROGRAM

This program describes the reaction between water and an extremely active metal such as sodium. The metal dissolves in water, evolving hydrogen gas and forming a basic solution.

Macroscopic Demonstration

The animation displays the following reactions:

Sodium reaction:

- a piece of sodium is dropped into a beaker of water.
- the metal dissolves on the surface of the water and a gas is evolved.
- the gas ignites spontaneously.
- after the sodium has completely dissolved a drop of litmus solution is added to the beaker, and the solution of dissolved sodium hydroxide changes to a blue colour.

Calcium reaction:

- a piece of calcium metal is dropped into a beaker of water.
- the metal sits on the bottom of the beaker and initially there is no visible reaction.
- the beaker is heated causing the calcium to dissolve and a gas is evolved.
- the gas explodes when exposed to a flame.
- the solution of dissolved metal hydroxide is tested with litmus, which turns the solution blue.

A general word equation is presented as a summary of the above reactions:

WATER + VERY ACTIVE METAL --> HYDROGEN + METAL HYDROXIDE

This is followed by a series of word equations relating to particular metals.

e.g. WATER + LITHIUM ---> HYDROGEN + LITHIUM HYDROXIDE

Microscopic demonstration

Introduction: The demonstration defines the shapes which will be used to represent water molecules and hydroxide ions.(Figure 5.5)

Metallic Structure: The structure of sodium atoms is illustrated followed by a brief description of crystal structure and metallic bonding.

Reaction between water and metal:

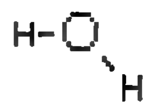
The main part of this program is an animation between water molecules and atoms of a very active metal. The water molecules strike the surface atoms of metal. (Figure 5.6) The valence electrons are transferred from the metal to the water molecule. The metal atoms are thereby converted into metal ions, and the water is converted into hydrogen gas and hydroxide ions. (Figure 5.7) This demonstration is carried out twice, the first time using sodium and the second time using calcium.

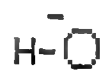
Summary of reaction: The net reaction of the metal atoms, and the molecules are graphically displayed.

Solvation of ions: Simplification regarding solvation of ions is discussed and hydrated metal ions are displayed.

This demonstration will display

following structures:-

WATER MOLECULE: 

HYDROXIDE ION: 

<u>Figure 5.5</u> <u>Activemetal Program</u>: definition of shapes of water molecules and hydroxide ions displayed in microscopic demonstration.

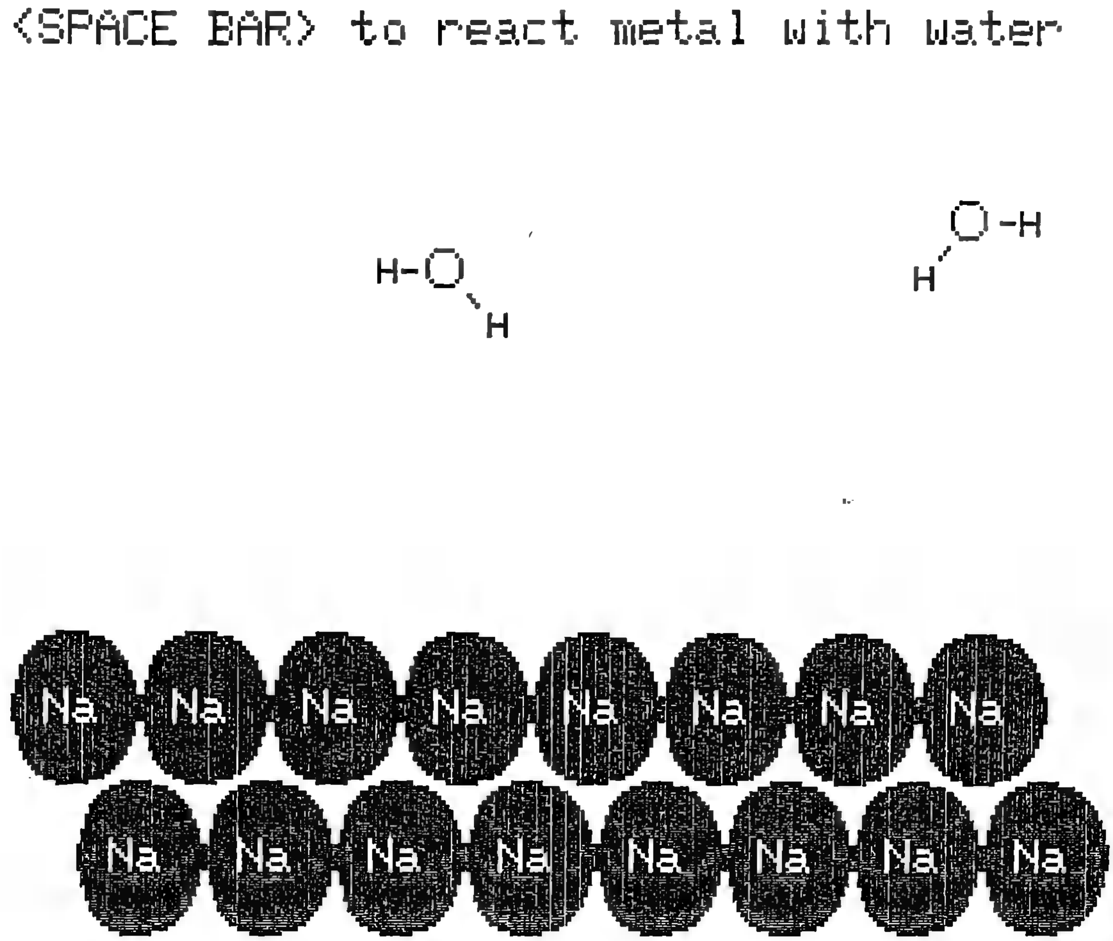


Figure 5.6 Activemetal Program: water molecules react with sodium metal.

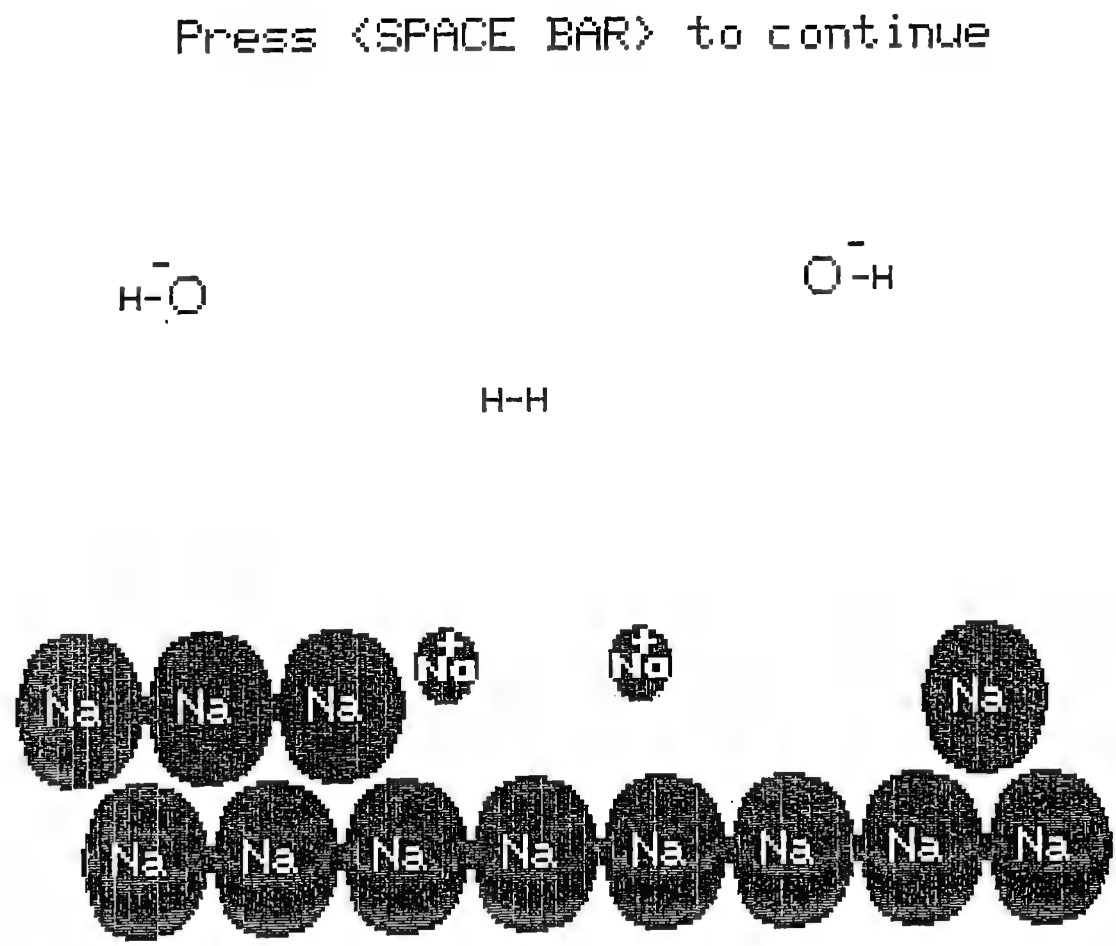


Figure 5.7 Activemetal Program: hydrogen molecules and hydroxide ions are formed as sodium metal loses electrons to water molecules.

5.1.4 ACID + CARBONATE PROGRAM

This program describes a reaction between a dilute acid and a carbonate. The carbonate dissolves and carbon dioxide gas is evolved.

Macroscopic Demonstration

A piece of calcium carbonate is dropped into a test tube of hydrochloric acid. The carbonate dissolves producing a gas. When tested with a flame, the gas causes it to be extinguished and when the gas is bubbled through limewater the solution becomes milky. (Figure 5.8)

A general word equation is presented as a summary of the above reaction:

ACID + CARBONATE --> CARBON DIOXIDE + WATER + A SALT

This is followed by a series of equations relating to specific acids and carbonates.

e.g.

SULFURIC ACID + MAGNESIUM CARBONATE --> CARBON DIOXIDE + WATER + MAGNESIUM SULFATE

Microscopic Demonstration

Introduction: The shapes used to represent the following species are defined: (Figures 5.9 (a) & (b))

water molecules
carbon dioxide molecules
calcium ions
sodium ions
carbonate ions

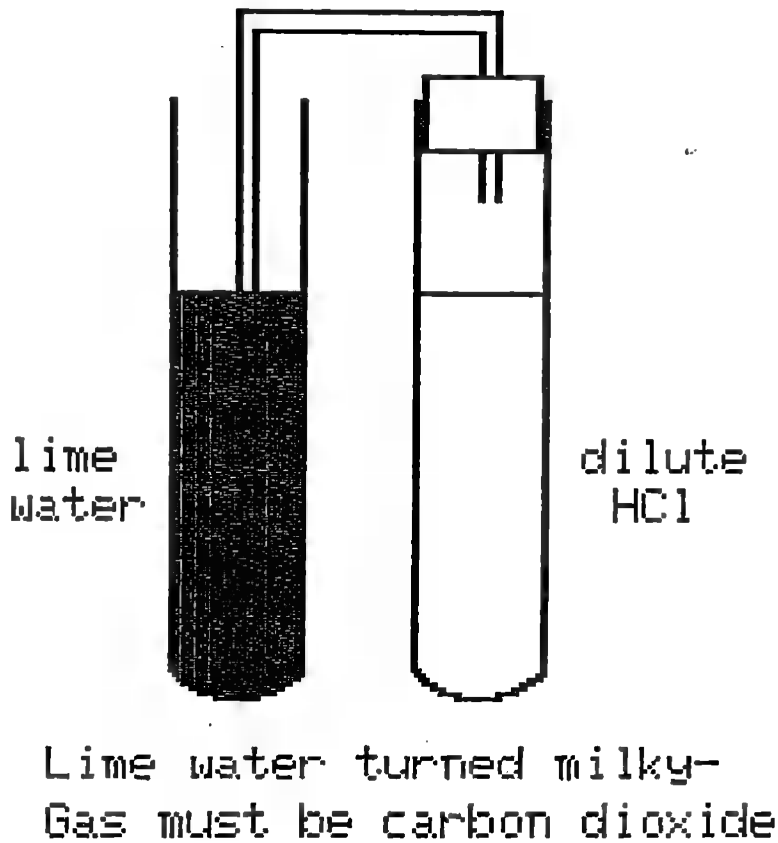
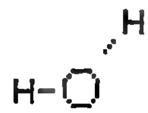
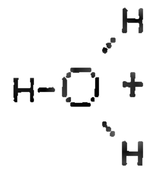


Figure 5.8 Carbonate Program: limewater turns milky.

This demonstration will display

the following structures:-

WATER MOLECULE: 

HYDRONIUM ION: 

CARBON DIOXIDE: 

Figure 5.9(a) Carbonate Program: definition of shapes displayed in microscopic demonstration.

Other structures displayed

in this demonstration:-

CALCIUM ION: 

SODIUM ION: 

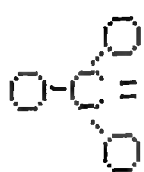
CARBONATE ION: 

Figure 5.9(b) Carbonate Program: definition of shapes displayed in microscopic demonstration.

Reaction between a carbonate and an acid.

Hydronium ions strike the surface atoms of a carbonate resulting the transfer of a protons from the hydronium ions to the carbonate ions. (Figures 5.10(a)-(d)) The hydronium ions are thereby converted to water, and the carbonate ions are converted to carbonic acid which decomposes to carbon dioxide and water. (Figure 5.11) This reaction is carried out with calcium carbonate and then with sodium carbonate.

Summary of reaction: The net reaction of hydronium ions and carbonate ions are graphically displayed.

The overall reaction is displayed using hydronium ions, and then compared with a simplified illustration using hydrogen ions.

Solvation of ions: The simplification regarding solvation of ions is discussed, and hydrated metal ions displayed.

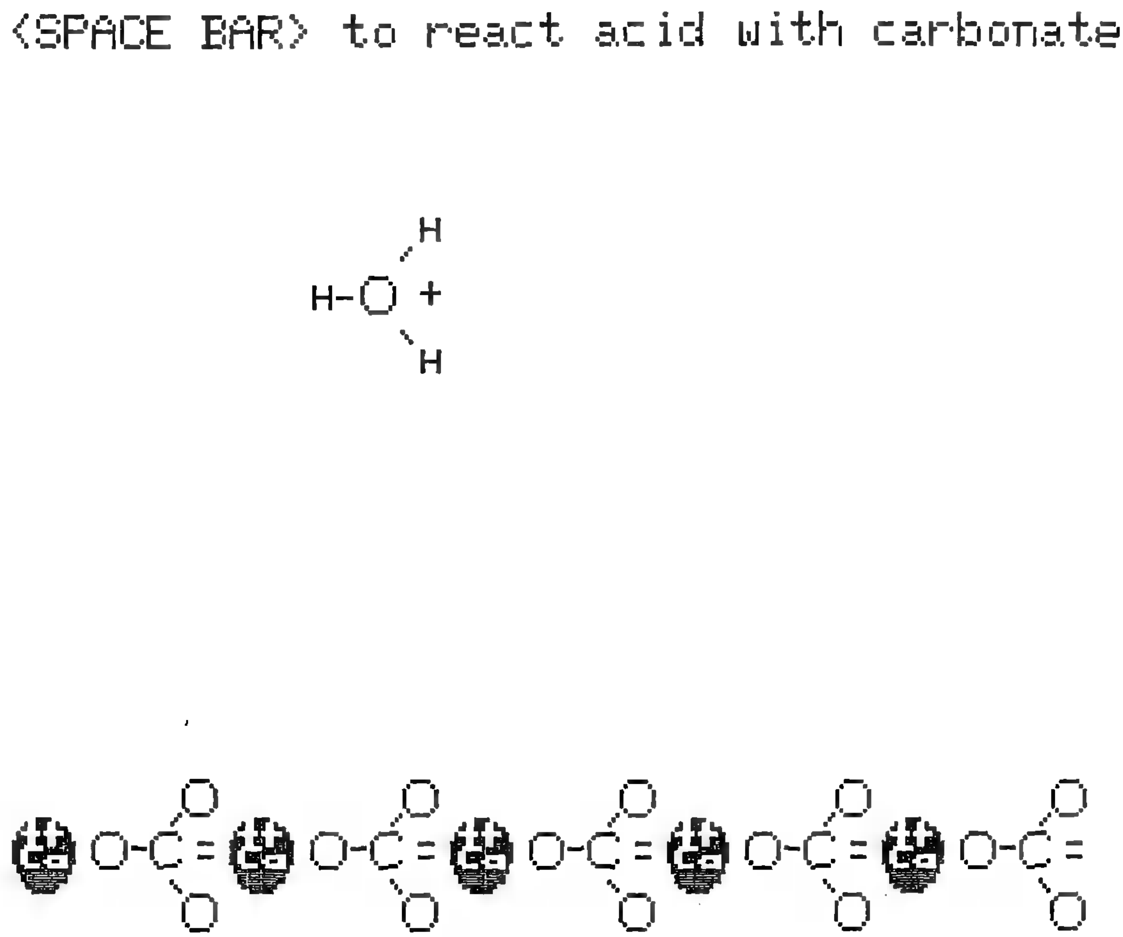


Figure 5.10(a) Carbonate Program: hydronium ion reacts with carbonate ion.

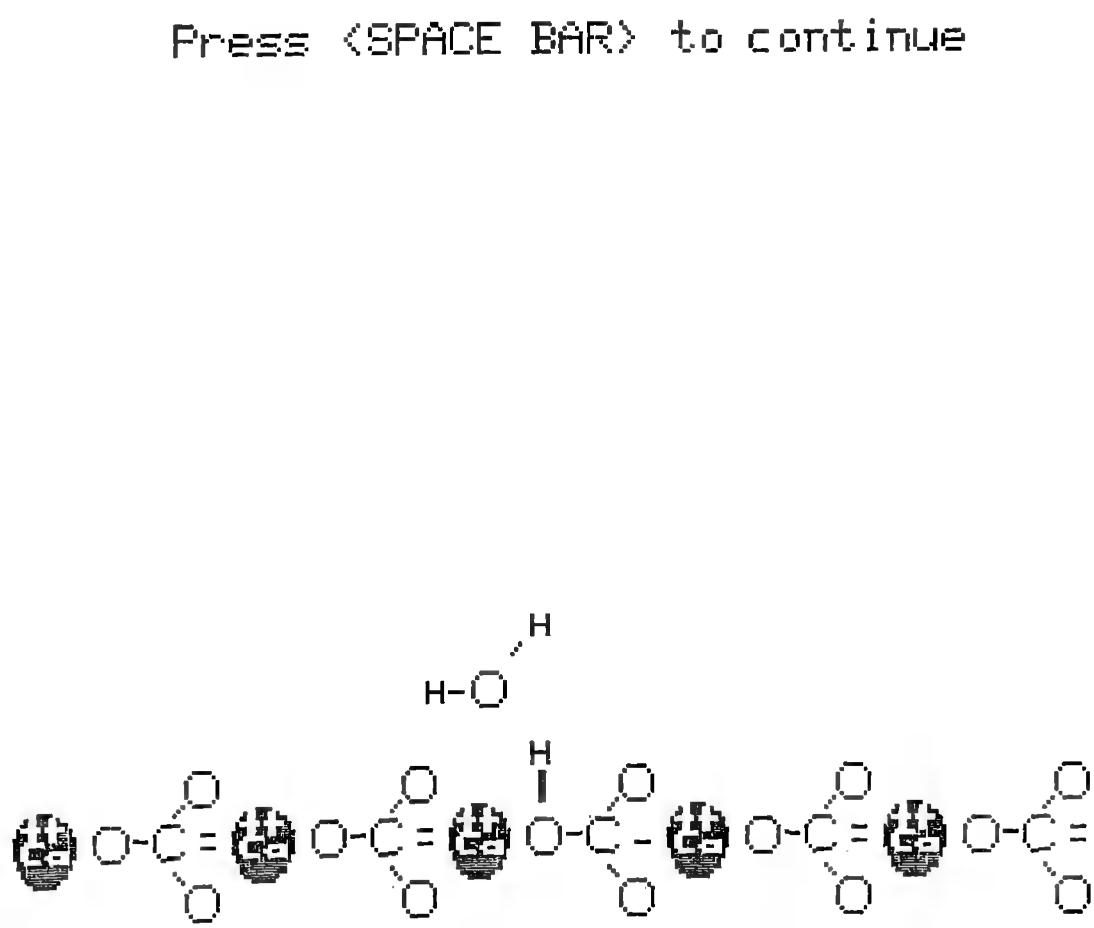


Figure 5.10(b) Carbonate Program: carbonate ion is converted to bicarbonate ion.

Press <SPACE BAR> to continue

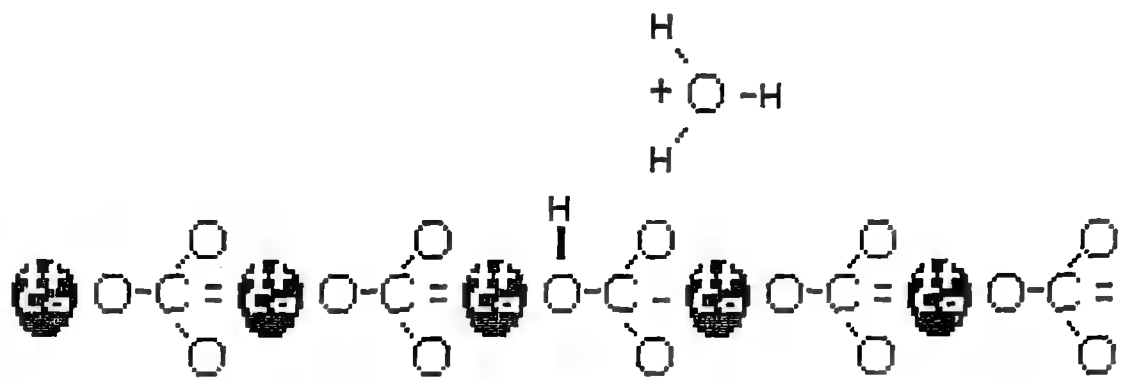


Figure 5.10(c) Carbonate Program: hydronium ion reacts with bicarbonate ion.

Press <SPACE BAR> to continue

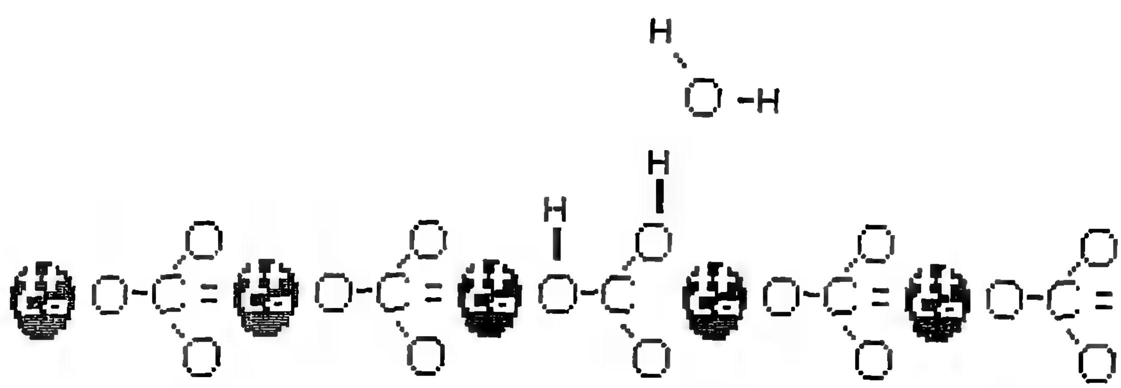
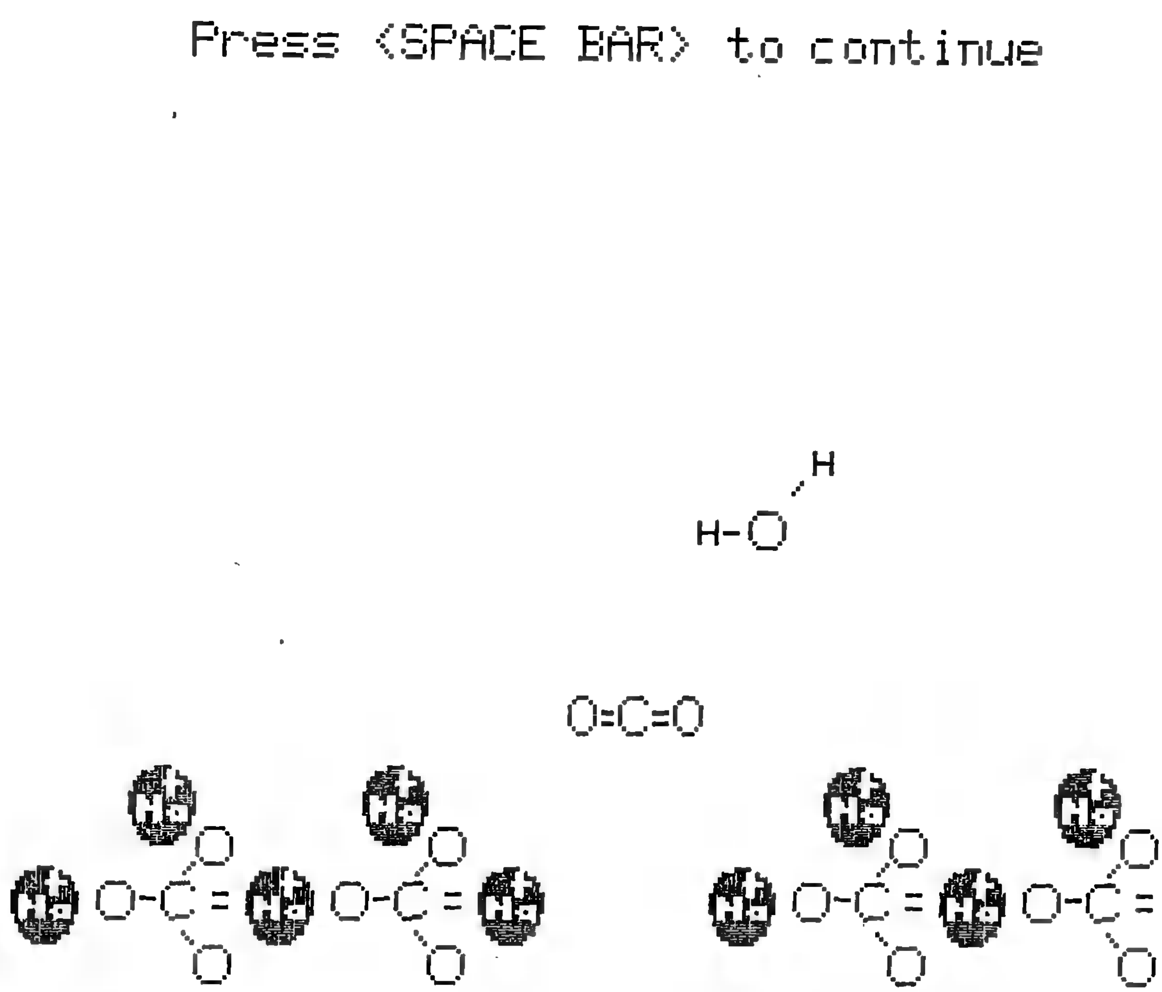


Figure 5.10(d) Carbonate Program: bicarbonate ion is converted to carbonic acid.



<u>Figure 5.11</u> <u>Carbonate Program</u>: carbon dioxide and water are formed from carbonic acid.

5.1.5 LITMUS PROGRAM

Litmus is a common indicator used to test the acidity of a solution.

Macroscopic Demonstration

An acidic solution is tested with blue and red litmus paper. The blue litmus turns red. This is repeated in basic solution in which the red litmus turns blue. (Figures 5.12 (a) & (b))
The litmus tests are repeated using litmus solution rather than litmus paper. The same colour changes are observed.

Microscopic Demonstration

Introduction: In this demonstration litmus is represented by a simple hexagonal shape. The blue form of litmus is the basic hexagon whereas the red form is a hexagon with an additional hydrogen ion.

Reaction between litmus and acidic solutions:

When blue litmus is placed in an acidic solution, the hydronium ions stike the surface of the litmus transferring a proton from the hydronium ion to the litmus. The hydronium ion is converted to water, and blue litmus is converted to red litmus. (Figure 5.13)
When red litmus is placed in an acidic solution there is no reaction.

Reaction between litmus and basic solutions:

When blue litmus is placed in a basic solution there is no reaction. When red litmus is placed in a basic solution, the hydroxide ions strike the surface of the litmus transferring a proton from the litmus to the hydroxide ions. The hydroxide ions are converted to water and the red litmus is converted to blue litmus. (Figure 5.14)

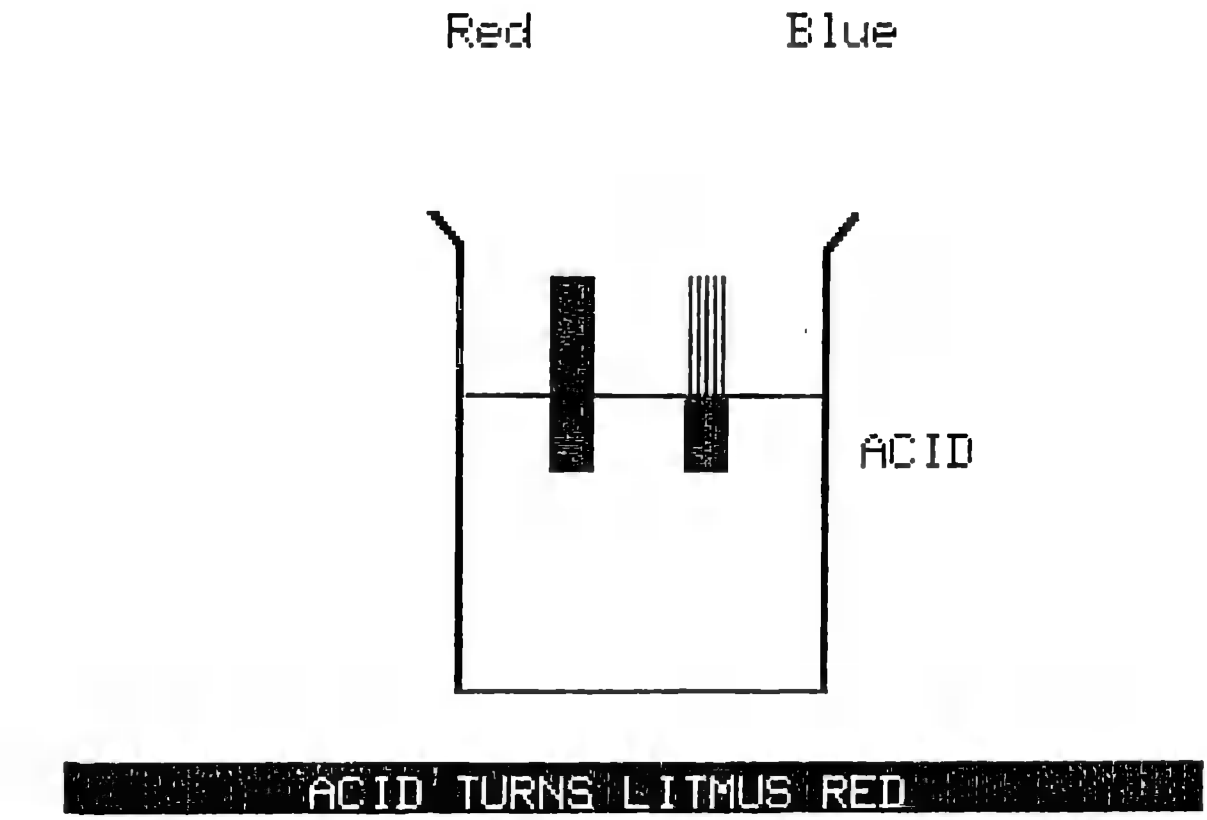


Figure 5.12(a) Litmus Program: litmus is red in acidic solution.

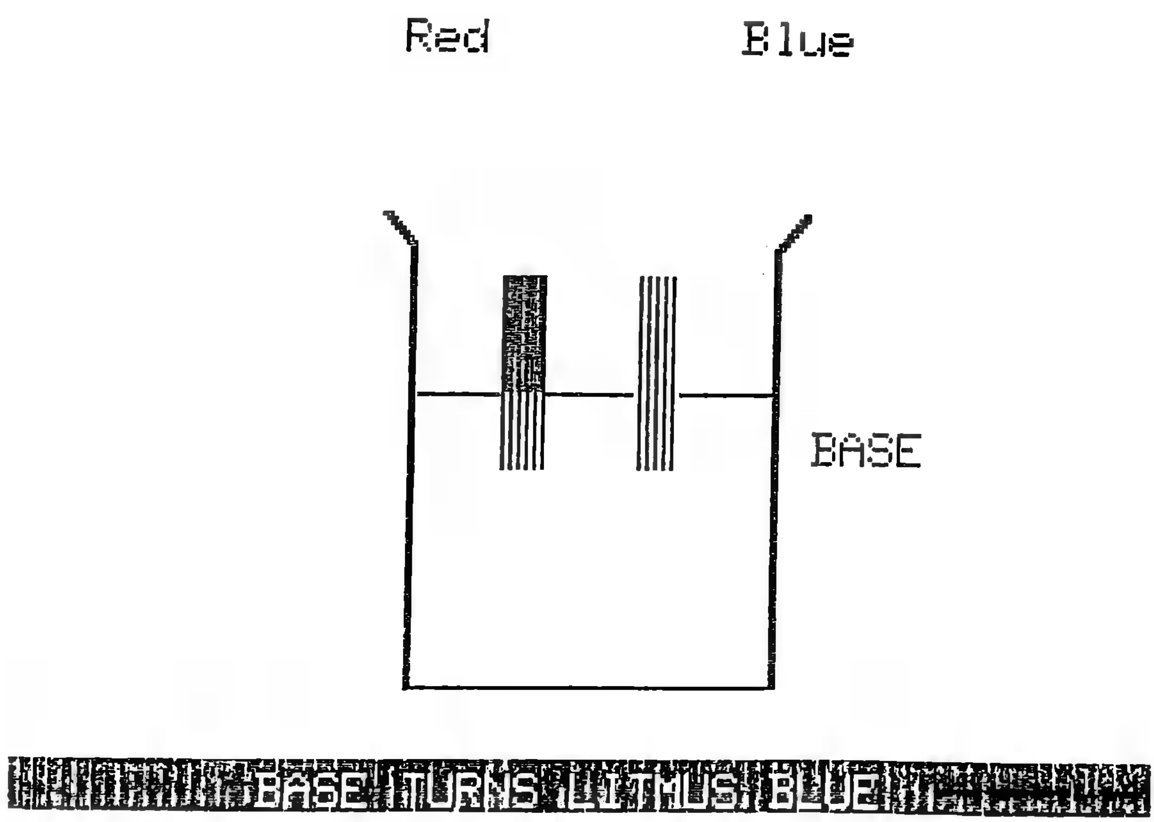


Figure 5.12(b) Litmus Program: litmus is blue in basic solution.

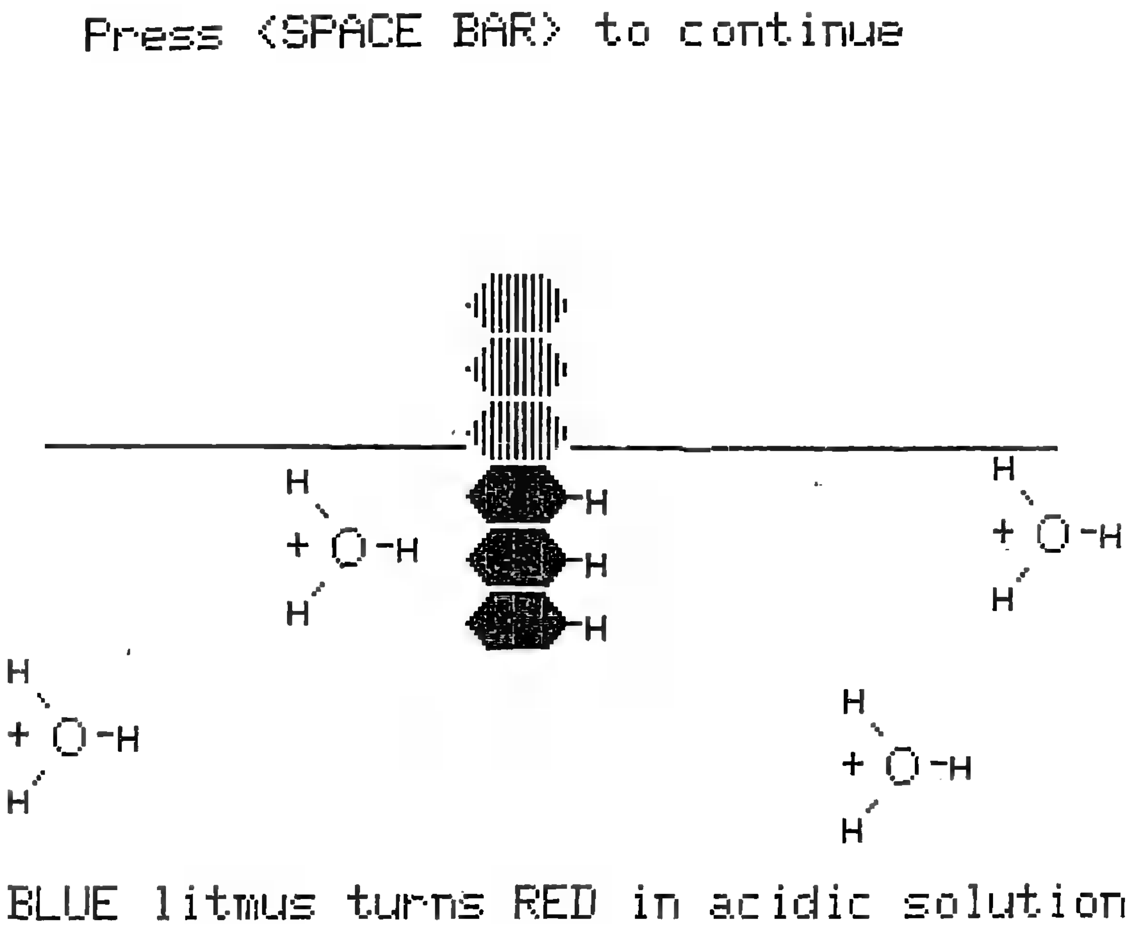


Figure 5.13 **Litmus Program:** blue litmus is converted to red litmus in acidic solution.

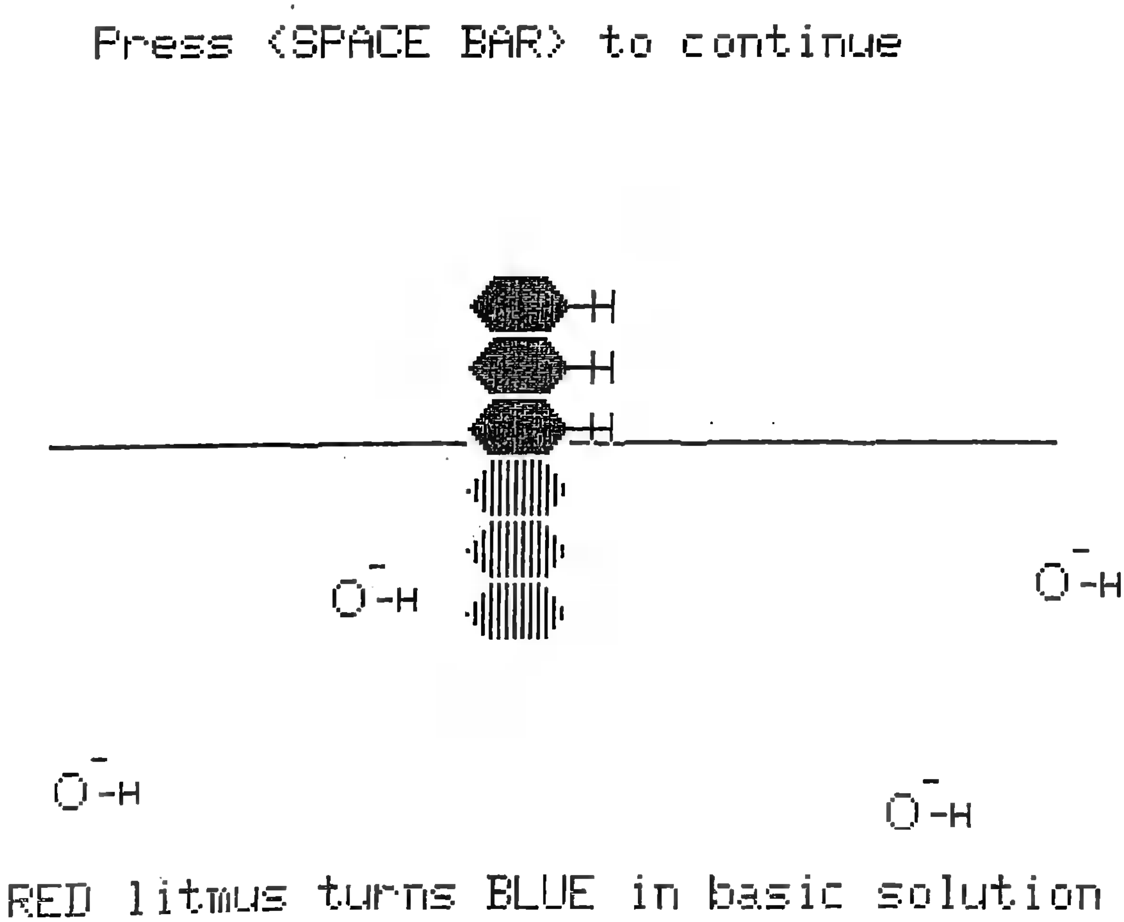


Figure 5.14 **Litmus Program:** red litmus is converted to blue litmus in basic solution.

5.2 OBJECTIVES OF THE MICRO/MACRO CHEM DEMONSTRATION PACKAGE

5.2.1 MACROSCOPIC DEMONSTRATIONS

The macroscopic representation of the reactions serves two purposes:

- firstly, it is a reminder of observations recorded in the laboratory.
- secondly, it is used to link the observable characteristics of a chemical reaction with the (unobservable) molecular changes involved.

5.2.2 MICROSCOPIC DEMONSTRATIONS

The microscopic representation of the reaction shows, in animated form, molecules undergoing a chemical change. The microscopic demonstrations illustrate that chemical reactions involve:

- collisions between reacting species;
- breaking and reforming chemical bonds; and
- rearrangement of atoms in molecules.

Further, the objective of each microscopic demonstration is to emphasise a number of reaction features. The specific objectives of each program are listed:

Acid and active metal demonstration

The objective of the acid and active metal microscopic demonstration is to focus on the following reaction features:

- the reaction involves electron transfer.
- the reaction occurs as a result of collisions between hydonium ions and the atoms on the surface of the metal.

- the metal loses electrons and is converted into metal ions.
- the metal dissolves because the metal ions are free to move around in the solution.
- the metal ions are much smaller than the metal atoms.
- hydronium ions gain electrons in the reaction.
- hydrogen gas is formed as a result of two hydronium ions gaining electrons from the metal.

<u>Water and very active metal demonstration</u>

The water and very active metal microscopic demonstration illustrates the following reaction features:

- the reaction involves electron transfer.
- the reaction occurs as a result of collisions between water molecules and the atoms on the surface of the metal.
- the metal loses electrons and is converted into metal ions.
- the metal dissolves because the metal ions are free to move around in the solution.
- the metal ions are much smaller than the metal atoms.
- the water molecules gain electrons.
- hydrogen gas is formed as a result of water molecules gaining electrons.
- the solution is basic due to the formation of hydroxide ions which are formed from the reacting water molecules.

<u>Acid + carbonate demonstration</u>

The objective of the acid and carbonate microscopic demonstration is to display the following features:

- the reaction involves proton transfer.
- the reaction occurs as a result of collisions between

hydronium ions in solution and the carbonate ions in the crystal.

- the carbonate ion gains one proton from a hydronium ion to form the bicarbonate ion.
- the bicarbonate ion gains a second proton from another hydronium ion to form a carbonic acid molecule.
- the carbonic acid molecule decomposes into carbon dioxide and water.
- the carbonate dissolves and the metal cation is released into solution.
- the reacting hydronium ions, lose a proton and are converted into water molecules.

Litmus reactions demonstration

The microscopic litmus demonstration emphasises the following features:

- litmus is a complex mixture, however the only difference between the red and blue forms of litmus is that the red litmus contains more protons than the blue form.
- red litmus is converted to blue litmus by reaction with a species which will remove a proton from the red litmus.
- blue litmus is converted to red litmus by reaction with a species which will donate a proton to the blue litmus.
- a litmus reaction involves a proton transfer.

5.2.3 WORKSHEETS

A series of worksheets has been designed to enable individual tutorial use of the MICRO/MACRO CHEM DEMONSTRATION PACKAGE. Each worksheet contains a number of questions which can be answered by viewing the demonstrations in this package. Utilization of the worksheet encourages the student to carefully analyse the reactions under consideration. The worksheets are listed in Appendix A.

5.3 PASCAL CODE FOR THE MICRO/MACRO CHEM DEMONSTRATION PACKAGE

The MICRO/MACRO CHEM DEMONSTRATION PACKAGE uses one customised library unit, Useful Unit, which is discussed in Section 2.2.3. In addition to the system files, the following object code files are included in the MICRO/MACRO CHEM DEMONSTRATION PACKAGE:

System.startup	(code for introduction section)
Menu.code	(code for main menu)
Metal.code	(code for active metal/acid program)
Activemetal.code	(code for very active metal/water program)
Carbonate.code	(code for acid/carbonate program)
Litmus.code	(code for litmus program)

The source code for all programs in this package are listed in Appendix E.

Graphics techniques employed in the MICRO/MACRO CHEM DEMONSTRATION PACKAGE utilize both "turtle" commands and bit-map transfer routines. Specific examples of these techniques are presented here.

5.3.1 MACROSCOPIC DEMONSTRATIONS

1. Shapes using "turtle" commands

The drawing of large objects such as test tubes and beakers employs "turtle" graphics. The relevant procedures are designed so that the apparatus may be of variable size and placed any where on the screen. The procedures designed to draw a beaker and a test tube will be discussed. These two procedures are

typical of those employing "turtle" commands.

Drawing a beaker

A beaker is required in the *Activemetal* and *Litmus* programs. PROCEDURE DRAWBEAKER simply uses turtle graphics to join together six points to represent a beaker. The procedure must be passed the coordinates of the bottom left hand corner of the beaker, the size of the beaker and its colour.

```
PROCEDURE DRAWBEAKER(X,Y,SIZE: INTEGER; COL: SCREENCOLOR);
(* x,y - coordinate of bottom left hand corner of beaker *)
(* size - width and height of beaker *)
VAR RIM:INTEGER;  (* size of lip on beaker*)
BEGIN
   RIM:=SIZE DIV 12;
   MOVECOL(X-RIM,Y+SIZE+RIM,COL);     (* position at top left corner*)
   MOVETO(X,Y+SIZE);                  (* draw rim on left hand side*)
   MOVETO(X,Y);                       (* draw left hand side*)
   MOVETO(X+SIZE,Y);                  (* draw base *)
   MOVETO(X+SIZE,Y+SIZE);             (* draw right hand side*)
   MOVECOL(X+SIZE+RIM,Y+SIZE+RIM,NONE); (* draw rim on right side*)
END;
```

Drawing a test tube

A test tube is required in the *Metal* and *Carbonate* programs. PROCEDURE DRAWTUBE draws a test tube with a curved base. DRAWTUBE must be passed the coordinate of the bottom left hand side of the test tube, the width and length of the tube and the level of solution relative to the base of the tube.

```
PROCEDURE DRAWTUBE(X,Y,WIDTH,SIZE,LEVEL:INTEGER;
                                          COL:SCREENCOLOR);
(* x,y - coordinate of bottom left hand side of testtube*)
(* width - width of testtube; size - length of testtube*)
(* level - relative height of solution in tube *)
VAR  SIXNTH, THIRTYSECOND, RWIDTH:REAL;
     POINT:INTEGER;
BEGIN
   (* variables required to produce curved bottom of testtube*)
   REALWIDTH:=WIDTH;
   SIXNTH:=REALWIDTH/16;
```

```
THIRTYSECOND:=REALWIDTH/32;

(* draw left hand side of testtube*)
MOVECOL(X,Y+SIZE,COL);
MOVETO(X,Y);

(*draw bottom curve of testtube*)
FOR POINT:=1 TO 7 DO
MOVETO(X+ROUND(SIXNTH*POINT),Y-
                ROUND((POINT+1)*THIRTYSECOND));
MOVETO(X+ROUND(SIXNTH*9),Y-ROUND(8*THIRTYSECOND));
FOR POINT:=10 TO 15 DO
MOVETO(X+ROUND(SIXNTH*POINT),Y-ROUND((17-
                           POINT)*THIRTYSECOND));

(*draw right hand side of testtube*)
MOVETO(X+WIDTH,Y);
MOVETO(X+WIDTH,Y+SIZE);

(*draw level of solution in testtube*)
MOVETO(X+WIDTH,Y+LEVEL);
MOVECOL(X,Y+LEVEL,NONE);
END; (*DRAWTUBE*)
```

2. Shapes using bit-map transfer

Smaller or detailed objects, or objects which require animation are often drawn using bit-map transfer techniques,

e.g. bubbles rising in solution
flame which moves across screen
piece of metal or carbonate which drops into solution
drop of litmus which falls into solution

Initializing shapes

In the *Metal* program, the flame, metal and bubbles are initialized in PROCEDURE INITSHAPES. The flame is represented by an 8 x 8 array of boolean; the metal pieces by three 6 x 16 arrays and the bubbles by a 6 x 8 array. INITSHAPES initializes the shapes in such a way that they are easily recognised in the text file, and are therefore easily altered.

```
PROCEDURE INITSHAPES;
VAR COL, ROW, MAXCOL, MAXROW: INTEGER;
BEGIN
 (* initialize metal shapes *)
 STR[5]:=' XXX   XXXXXX  ';
 STR[4]:=' X XX XXX   XXX';
 STR[3]:='X   X          X';
 STR[2]:='X              X';
 STR[1]:='X         XXX  X';
 STR[0]:=' XXXXXXX  XXX ';
 MAXROW:=5; MAXCOL:=15;
 FOR ROW:=0 TO MAXROW DO
     FOR COL:= 0 TO MAXCOL DO METAL[1]:=STR[ROW][ROW,COL+1]='X';

 STR[5]:='                ';
 STR[4]:='   XX     XXX   ';
 STR[3]:=' X   X  XX   X  ';
 STR[2]:='  X    XX    X  ';
 STR[1]:='   X      XXXX  ';
 STR[0]:='    XXXX        ';
 FOR ROW:=0 TO MAXROW DO
     FOR COL:= 0 TO MAXCOL DO METAL[2]:=STR[ROW][ROW,COL+1]='X';

 STR[5]:='                ';
 STR[4]:='                ';
 STR[3]:='     X    X     ';
 STR[2]:='    X X XX X    ';
 STR[1]:='     X X XX     ';
 STR[0]:='      XXX       ';
 FOR ROW:=0 TO MAXROW DO
     FOR COL:= 0 TO MAXCOL DO METAL[3]:=STR[ROW][ROW,COL+1]='X';

 (* initialize bubble shape *)
 STR[5]:='  XX  ';
 STR[4]:=' X  X ';
 STR[3]:='X    X';
 STR[2]:='X    X';
 STR[1]:=' X  X ';
 STR[1]:='  XX  ';
 MAXCOL:=7;
 FOR ROW:=0 TO MAXROW DO
      FOR COL:= 0 TO MAXCOL DO BUBBLE:=STR[ROW][ROW,COL+1]='X';

 (* initialize flame shape *)
 STR[7]:='   XX   ';
 STR[6]:='   XX  ';
 STR[5]:='  X  X ';
 STR[4]:=' X    X ';
 STR[3]:='X      X';
 STR[2]:='X      X';
 STR[1]:=' X    X ';
 STR[0]:='   XX   ';
```

```
    MAXROW:=7;
    FOR ROW:=0 TO MAXROW DO
        FOR COL:= 0 TO MAXCOL DO FLAME:=STR[ROW][ROW,COL+1]='X';
  END; (*INITSHAPES*)
```

Movement of shapes

Movement of objects is achieved by drawblocking at the current position and then drawblocking at new position using Exclusive Or mode (XOR). In the *Metal* program, a piece of metal is dropped into a test tube by PROCEDURE DROPMETAL:

```
  .
  .
  REPEAT
    DRAWBLOCK(METAL[1],2,0,0,16,6,METALX,Y,MODE); (*erase at current
    Y:=Y-10;                                                position*)
    DRAWBLOCK(METAL[1],2,0,0,16,6,METALX,Y,MODE); (*redraw at new
    DELAY(20);                                              position*)
  UNTIL Y<= BOTTOM;
```

The metal appears to dissolve as the shape of the metal becomes smaller. PROCEDURE SWAPMETAL is called to replace the current metal shape with a smaller one:

```
  PROCEDURE SWAPMETAL(VAR CURRENT: INTEGER);
  BEGIN
    DRAWBLOCK(METAL[CURRENT],2,0,0,16,6,METALX,METALY,MODE);
                                                     (*erase old*)
    CURRENT:=CURRENT+1;
    DRAWBLOCK(METAL[CURRENT],2,0,0,16,6,METALX,METALY,MODE);
  END; (*SWAPMETAL*)                               (*display new*)
```

5.3.2 MICROSCOPIC DEMONSTRATIONS

The shapes of atoms and molecules required in the microscopic demonstrations are more complex than those in the macroscopic demonstrations. Drawing these shapes is handled solely by the drawblock method. Techniques employed to represent and animate metal atoms and ions are discussed:

Initializing metal atoms

In the *Metal* and *Activemetal* programs the surface layer of metal atoms is represented by two rows of spheres. (Figures 5.3, 5.4, 5.6 & 5.7)

Depending on the metal being considered the initials Mg (magnesium), Ni (nickel), Na (sodium), Ca (calcium) are displayed on each atom. At the start of each program a two dimensional array of Boolean is initialized so as to represent a sphere:

```
PROCEDURE INITMETAL;
CONST MAX=32;
VAR STR: ARRAY [1..MAX] OF STRING;
    ROW: INTEGER;

        PROCEDURE INIT(ROW:INTEGER;VAR BITS:BIGSHAPE; S:STRING);
        VAR COL: INTEGER;
        BEGIN
           FOR COL:1 TO MAX DO BITS[ROW,COL]:=STR[COL]='X';
        END;

  BEGIN (*INITMETAL*)
     STR[1]:='            XXXXXXX             ';
     STR[2]:='         XXXXXXXXXXXXX          ';
     STR[3]:='      XXXXXXXXXXXXXXXXXX        ';
     STR[4]:='    XXXXXXXXXXXXXXXXXXXXX       ';
     STR[5]:='  XXXXXXXXXXXXXXXXXXXXXXXX      ';
     STR[6]:='  XXXXXXXXXXXXXXXXXXXXXXXXX     ';
     STR[7]:='  XXXXXXXXXXXXXXXXXXXXXXXXXX    ';
     STR[8]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXX    ';
     STR[9]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXX    ';
    STR[10]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXX   ';
    STR[11]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
    STR[12]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
    STR[13]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    STR[14]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
    STR[15]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    STR[16]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    FOR ROW:=1 TO MAX DIV 2 DO
     BEGIN
       INIT(ROW,ATOM,STR[ROW]);
       INIT(MAX+1-ROW,ATOM,STR[ROW]);
     END;
  END; (*INITMETAL*)
```

Displaying metal atoms

Procedure drawblock is used to draw each metal atom (sphere), it is then necessary to superimpose the appropriate symbol onto each atom. This is achieved in PROCEDURE DRAWMETAL by the following statements:

```
DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
WSTAT(X+8,Y+12,SYMB);
```

Where mode is equal to 6 and SYMB is a string containing appropriate symbol for each metal.

The metal atoms do not move, but are erased and replaced by metal ions at the appropriate part of the program. In *Metal* the shape of the cation is represented by, CATION, a 16 x 16 array of boolean. The position and direction of movement of the cation is stored in record METALION:

```
TYPE ION= RECORD
            X,Y,                 (* current position of cation *)
            DX,DY : INTEGER; (*movement in x & y direction*)
          END;
VAR  METALION: ION;
```

Swapping metal atoms for metal ions

In the *Metal* program, PROCEDURE IONIZE initializes the position of the cation and replaces a metal atom with a cation:

```
PROCEDURE IONIZE(VAR METX,METY: INTEGER);
BEGIN
   DRAWBLOCK(ATOM,4,0,0,32,32,METX,METY,4);   (*erase metal atom*)
  WITH METALION DO
     BEGIN
        X:=METX+8; Y:=METY+8;     (*initialize coordinates of metal ion*)
         DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);  (*display cation*)
     END;
END; (*IONIZE*)
```

Moving metal ions

In *Metal* program movement of the cation is carried out by

PROCEDURE MOVEION:

```
PROCEDURE MOVEION;
BEGIN
  WITH METALION DO
    BEGIN
      DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE); (*erase at current
                                                          position*)
      X:=X+DX;   Y:=Y+DY;                     (*calculate new
                                                          position*)
      DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE); (*display at new
                                                          position*)
    END;
END; (*MOVEION*)
```

6. EVALUATION

The three packages developed in this project: ACID/BASE TITRATION PACKAGE; SALT TITRATION PACKAGE and MICRO/ MACRO CHEM DEMONSTRATION PACKAGE have been trialled in the following chemistry classes:

Chemistry Bridging Course - Wollongong University.
First Year Remedial Chemistry - Wollongong University.
Day Matriculation Chemistry - Wollongong College of TAFE.
University Preparation Chemistry Course - Wollongong College of TAFE.

ACID/BASE TITRATION PACKAGE

An early version of the ACID/BASE TITRATION PACKAGE was tested in the chemistry classroom in 1985 (Day Matriculation Chemistry at Wollongong College of TAFE) and resulted in several improvements and extensions to the package. The final version of the ACID/BASE TITRATION PACKAGE has been assessed over the last two years in the Day Matriculation Chemistry and University Preparation Chemistry classes . In addition the package has been utilized for remedial tuition of first year chemistry students at Wollongong University.

In all classroom situations, the programs were used both as visual aids to complement chemistry lectures and also by students for individual tutorial lessons. The package was found to provide excellent visual displays enhancing lectures on acid/base titrations. All students were extremely enthusiastic towards viewing and using the programs. Students found the

programs easy to use, informative and highly motivating. Unfortunately, since most students had only limited experience with educational computer programs, enthusiastic response may also be partially attributable to the novelty of using computers.

Thorough testing of this package has enabled a number of minor programming "bugs" to be discovered and corrected. Assessment during last year, has not revealed further problems.

SALT TITRATION PACKAGE

Evaluation of this package has not been as extensive as the ACID/BASE TITRATION PACKAGE, however all student responses have been extremely favourable. The program has principally been utilized in the lecture situation providing excellent graphical illustrations of the concepts involved in the titration of acidic salts. During the current year, a number of TAFE students have attempted the assignment programs in the package. All students exposed to the programs considered them to be most valuable in mastering the titration concepts considered.

Remedial chemistry students at Wollongong University found the programs involving titration of a mixture of sodium carbonate and sodium bicarbonate particularly useful, as such titrations are part of a required laboratory experiment in first year chemistry. Students commented that the programs readily illustrated the concepts involved in this experiment.

MICRO/MACRO CHEM DEMONSTRATION PACKAGE

This package has been used in a number of TAFE chemistry classes this year, and was highly successful in illustrating the molecular features of several fundamental chemical reactions. The programs were firstly utilized as visual aids within chemistry lectures and then students viewed the programs in small groups, attempting to complete the complementary worksheets. Again, student response was extremely positive - students generally commented that the animation of molecules helped them to visualise the chemical reactions involved. A most common student response was that the programs provided them with a clear picture of why certain products were formed.

Conclusion

Although only a very general qualitative assessment of the packages has been made, it is clear that all packages developed in this research project provide a valuable and versatile aid to teaching specified concepts in chemistry.

BIBLIOGRAPHY

Adams,K.A.H.,Storer,A.C. & Cornish-Bowden,A. "Enzyme Kinetics Calculations - The Direct Linerar Plot Procedure", *J.Chem.Educ.*, **61**,6, (1984).

Alpert,D. & Bitzer, D.L. "Advances in Computer-Based Education", *Science*, **167**, 3925, (1970).

Armanious,M. & Shoja,M. "Analysis of the Band Spectrum of I2 Using Apple II", *J.Chem.Educ.*, **63**,7, (1986).

Barrow,G.M. "Computer-Based Studies for Physical Chemistry", *J.Chem.Educ.*,**57**,10, (1980).

Bath,D.A. & Hughes,B.G. "Computerized Quizzes with Instant Grading and Response Analysis", *J.Chem.Educ.*, **60**,9, (1983).

Bauder,D. "Haber-Tech:A Simulation of the Industrial Synthesis of Ammonia", COMPress, A Division of Wadsworth, Wentworth, (1985)

Bendall,V.I. "A Programming Utility for Animation", *J.Chem.Educ.*, **64**, 3, (1987).

Biro,P. & Sharpely,B. "What Chemical is that? An inorganic chemistry data base", Prologic, Abbotsford, (1986).

Bishop,M., Frinks,S., & Meier,A.M. "Applications of Computer Graphics in the Sudy of Polymer Configurations", *J.Chem.Educ.*, **63**,9, (1986).

Bork,A. "Computers and the Future:Education", *Comput.Educ.*, **8**,1 (1984).

Bourque D.R. and Carlson G.R., "Hands-On versus Computer Simulation Methods in Chemistry", *J.Chem.Educ.*, **64**,3 (1987).

Brandwood,J. & Taylor, A. "Symbols to Moles" ; Five Ways Software; Heinemann Computers in Education, London (1982).

Breneman,G.L. "The Impact of Micrographics", *J.Chem.Educ.*, **58**,12, (1981).

Brownawell,M.L. & Filippo,J.S. "A Program for the Synthesis of Mass Spectral Isotopic Abundances", *J.Chem.Educ.*, **59**,8, (1982).

Butler,W. "Computer Aided Instruction:General Chemistry", John Wiley & Sons;New York, (1983).

Coe,D.A. "Using a Spreadsheet to Calculate the Fugacity of a van de Waals Gas", *J.Chem.Educ.*, **64**,2, (1987).

Cornelius,R. "Concentrated Chemical Concepts", John Wiley & Sons; New York, (1983).
Cornelius,R.D. & Norman,P.R. "A Simple and Inexpensive pH-stat and Autotitrator Based on the Apple II Plus Computer", *J.Chem.Educ.*, **60**,2, (1983).
Dessy,R.E. "Chemistry and the Microcomputer Revolution", *J.Chem.Educ.*, **59**,4, (1982).
Dessy,R.E. & Starling,M.K., "Information Retrieval and Laboratory Management", *Anal.Chem.*, **51**,9, (1979)
Draper,R.D. & Penfold,B.R. "Nuclear Magnetic Resonance Interpretation with Graphics", *J.Chem.Educ.*, **61**,9, (1984).
Farrell,J.J. "Colour Images of Molecules", *J.Chem.Educ.*, **64**,3 (1987).
Feng,A. & Moore,J. "Exploring Chemistry by Computer:KC? Discoverer", *J.Chem.Educ.*, **63**,4, (1986).
Flash,P. "An Organic Synthesis Program for Allied Health Chemistry", *J.Chem.Educ.*, **64**,3, (1987).
Flash,P.J. "Graphics Drill and Game Programs for Benzene Synthesis", *J.Chem.Educ.*, **62**,11, (1985).
Fleisher,P. "Chemical Elements", Hartley Courseware, Dimondale (1984).
Frazin,J. & Partners "Chemistry: Acids and Bases", Encyclopaedia Britannica Educ.; Chicago, (1983).
Geanangel,R. & Beneke,J. "Basic Programs for Calculating Overlap Integrals", *J.Chem.Educ.*, **63**,9, (1986).
Gerhold, G., "Computers and the High School Chemistry Teacher", *J.Chem.Educ.*, **62**,3, (1985).
Gilbert,D.D.,Mounts,T.T. & Frost,A.A. "Computer-Graphics Emulation of Chemical Instrumentation: Absorption Spectrophotometers", *J.Chem.Educ.*, **59**,8, (1982).
Gouge,E.M. "Employing Data Management Software for the Production and Searching of Customized Mass Spectral Libraries", *J.Chem.Educ.*, **61**,9, (1984).
Holdsworth,D.K. "Conductivity Titrations - A Microcomputer Approach", *J.Chem.Educ.*, **63**,1, (1986).
Horst,K.E. & Dowden,E. "Collect Chem Data by Computer", *Science Teacher*, 53,8, (1986).
Howbert,J.J.,Lilly,E., & Co. "Molecular Animator", COMPress, A Division of Wadsworth, Inc., (1985).

Hull,L.A. "Animated 3-D Graphical Display of Line Drawings of Molecules", *J.Chem.Educ.*, **60**,2, (1983).

Johnson,K.J. "Simulation and Data Reduction Programs", *J.Chem.Educ.*, **57**,6, (1980).

Joshi,B.D. "An Interactive,Screen-Oriented,General Linear Regression Program", *J.Chem.Educ.*, **62**,11, (1985).

Klopfer,L.E. "The Coming Generation of Tutoring Software", *Science Teacher*, **53**,8, (1986).

Kolodny,N.H. and Bayly,R. "Enhancing the Laboratory Experience in Introductory Chemistry with Apple-based pre-lab Quizzes", *J.Chem.Educ.*, **60**,10, (1983).

Kubach,C. "Illustration of Quantization and Perturbation Theory Using Microcomputers", *J.Chem.Educ.*, **60**,3, (1983).

Luibrand,R.T. & Smith,V.,"Enhance Your Spreadsheet Capabilities", *J.Chem.Educ.*, **64**,3, (1987).

Moore J.W. & Moore,E.A. "Powwow:The Future of Microcomputers in Chemical Education", *J.Chem.Educ.*, **61**,11, (1984).

Moore,J., Miles,P.,Rasmussen,M.,Hartman,K.,Barker,P., "Inexpensive Computerized Experiments", *J.Chem.Educ.*, **63**,4, (1986).

Moore, J.W. "Editors' Note: Availability of Computers", *J.Chem.Educ.*, **64**,3, (1987).

Nakano,H., Sangen,O.,Yamamoto,Y.,"Drawing of Ball and Stick Type Molecular Models with Hidden Line Elimination", *J.Chem.Educ.*, **60**,2, (1983).

Newmark,R.A. "Dynamic NMR Spectra of Two-Spin Systems", *J.Chem.Educ.*, **60**,1, (1983).

Olmsted,S.L. & Olmsted,R.D.,"Pre-lab Studies for General Organic and Biological Chemistry", John Wiley & Sons; New York, (1983).

Pankuch,B.J. "Chemical Bonding Simulation", *J.Chem.Educ.*, **61**,9, (1984).

Papert,S. "Mindstorms: Children, Computers and Powerful Ideas", Basic Books; New York (1980).

Rhodes,B. "Protein Graphics on the Commodore 64 Microcomputer", *J.Chem.Educ.*, **63**,12, (1986).

Rinehart,F.P. "The Chemistry Tutor", Wiley Education Software; Somerset, (1985).

Rusch,P.F. "Introduction to Chemical Information Storage and Retrieval", *J.Chem.Educ.*, **58**,4, (1981).

Russell,A.A. "From Videotapes to Videodiscs", *J.Chem.Educ.*, **61**,10, (1984).

Ryan,J. "Find-the-Pairs", *J.Chem.Educ.*, **63**,7, (1986).

Schibeci,R.A. "Educational Software:Good,Bad or Indifferent", *Australian Science Teacher's Journal*, **32**, 3, (1985).
Settle,F.A. "The Application of Expert Systems in the General Chemistry Laboratory", *J.Chem.Educ.*, **64**,4, (1987).
Shumway,S. & Elliott,L. "Science Toolkit", Broderbund (1985).
Simpson,B. " Computer Simulation of Chemical Equilibrium", *Australian Science Teacher's Journal*, 32,1, (1986).
Smith,S.G. "Introduction to Organic Chemistry", COMPress, A Division of Wadsworth; Wentworth, (1981).
Smith,S.G. "Computer-Assisted Instruction on a Microcomputer", *J.Chem.Educ.*,**61**,10, (1984).
Smith,S.G.,Chabay,R., & Kean,E.,"Introduction to General Chemistry", COMPress, A Division of Wadsworth; Wentworth, (1985).
Starkey,R. "NMR Simulation Program for the ZX81 Computer", *J.Chem.Educ.*, **63**,7, (1986).
Suder,R. "Boyle's Law Simulation", *J.Chem.Educ.*, **60**,9, (1983).
Suetler,C.H.,& Hill,D.,"Simulation of Chemical and Enzyme Kinetics Experiments", *J.Chem.Educ.*, **58**,12, (1981).
Suetler,C.H.,Morris,A.J. & Hill,D.,"Scintillation Spectometry: Microcomputer Simulation", *J.Chem.Educ.*,**58**,12, (1981).
Tirri,L.J. & Jurutka,P.W. "Computer Simulated Metabolism", *J.Chem.Educ.*, **63**,12, (1986).
Traeger,J.C. "Microcomputer-Controlled Flash Photolysis Experiment", *J.Chem.Educ.*, **58**,12, (1981).
Waught,M. "Diagnostic tests by microcomputer", *Science Teacher*, **54**,3, (1987).
Whisnant,D.M. "Gas Chromatography Simulation for TRS-80", *J.Chem.Educ.*, **60**,1, (1983).
Whisnant,D.M. "Scientific Exploration with a Microcomputer: Simulations for Nonscientists", *J.Chem.Educ.*, **61**,7, (1984).
Wood,J.A. "Information Storage and Retrieval Using a Microcomputer", *J.Chem.Educ.*, **63**,7, (1986).
Zaks,R. "Introduction to Pascal Including UCSD Pascal", Sybex; (1981).

CONTENTS

Volume II

ACID / BASE TITRATIONS

Practical experience in carrying out titrations is obviously essential to the understanding of acid / base titrations. However there is rarely the time, or resources, for students to perform more than a few simple titrations. These programs provide the opportunity to simulate a wide variety of titrations.

These titration programs may be used by the teacher as a visual teaching aid to illustrate the relationships between the various titration variables and pH. Concepts such as choice of indicator, and effect of solution strength on end point, which are usually difficult for the student to understand by most teaching methods are readily illustrated and explained by these programs.

However, these programs have a much greater potential, as they allow the student to discover these concepts through scientific investigation. By carrying out simulated titrations the student will be able to derive the relationships between acids, bases and pH. The worksheets provided guide the student through the investigation of each concept so that relationships may be quickly discovered.

The user should find the instructions within the programs themselves sufficiently detailed. However written instructions have been provided.

ACID/BASE TITRATIONS consists of four programs which are briefly described on the following pages. It is suggested that the four programs are viewed in the order described.

IMPORTANT:
It is necessary for the TEACHER to refer to the ASSIGNMENT INSTRUCTIONS in order to organise student assignments. This manual contains the solutions to all the assignments.

(1) TITRATION OF ACIDS & BASES

This program allows the student to select the concentration of acid and base. An ideal indicator changes colour exactly at the equivalence point. It is necessary for the student to select the particular acid and base. The following solutions are available:

Strong acids - hydrochloric, nitric, perchloric.
Strong bases - sodium hydroxide, potassium hydroxide.
Weak acids - acetic, hydrocyanic, hydrofluoric, benzoic
(or input pKa.)
Weak base - ammonia, pyridine (or input pKb.)
Diprotic acids - sulphuric, carbonic, oxalic, tartaric
(or input pK1 & pK2)

Investigation of various titration variables are possible.
eg. The student is encouraged to ask questions such as:
What is the effect of changing concentration of acid or base or both?
What is the effect of changing the strength of either the acid or base?

A series of titrations may be carried out repeatedly. Each time the student can alter one variable to observe the results. If option to retain pH graph is selected then a graphical comparison is available.

(2) TITRATION OF ACIDS & BASES USING INDICATORS

Whereas the previous program provided an ideal indicator, the student may now select the indicator to be used with each simulated titration. The student is encouraged to investigate the suitability of different indicators for different titrations based on the pH range in which they change colour. (Other aspects relating to the suitability of indicators such as ease of colour detection have not been considered.) This program illustrates the effect of an indicator on the end point, which is awkward to display by any other method.

(3) QUIZ

The student may carry out simulated titrations in order to determine the concentration of a solution which has been selected by the computer. The computer will inform the user as to the accuracy of his titration results.

(4) ASSIGNMENT

This program is designed so that the student can determine the concentration of a sample solution. The nature and concentration of this solution depends on the assignment number entered at the start of this program. This assignment number must be provided by the teacher. It is essential that the teacher refer to the instructions for this program and allocate appropriate assignments for each student. Since the program does not inform the student of the correct concentration of the sample solution, the student can then submit the results of his titration for marking.

Although useful for student assessment, this is not the aim of the ASSIGNMENT program.

The most important aspect of the QUIZ and TITRATION programs is that they will enable the student to develop the necessary thinking skills required to carry out any acid/ base titration. When presented with a sample solution the student must select certain conditions such as concentration of standard solution. Through repeated titrations the student must develop a method, of altering these conditions, so that the concentration of the sample solution may be determined.

INSTRUCTIONS - TITRATION OF ACIDS AND BASES

Step 1: From Main Menu select option "1" .

MAIN MENU

Titration of acids & bases	**.......(1)**
Titration of acids & bases using indicators	**.......(2)**
Titration Quiz	**.......(3)**
Titration Assignment	**.......(4)**
Quit	**.......(Q)**
Select option	**....................(1)**

Step 2: From Titration Menu select appropriate titration.

TITRATION MENU

Strong acid / strong base	**......(1)**
Weak acid / strong base	**......(2)**
Weak base / strong acid	**......(3)**
Diprotic acid / strong base	**......(4)**
Quit - back to main menu	**......(Q)**
Select titration	**............( )**

Step 3: Select which acid and base is to be used.

Select weak acid to be used in titration

Acetic acid	**......(1)**
Hydrocyanic acid	**......(2)**
Hydrofluoric acid	**......(3)**
Benzoic acid	**......(4)**
Input pKa of weak acid	**......(5)**
Select option	**...............( )**

Step 4: For all titrations the following information must be entered:

	Sample input
Concentration of acid (in molarity) :	1.0
Concentration of base (in molarity) :	1.0
Solution in flask: either acid or base (A/B):	A
Volume of acid in flask (in mL) :	25.0
Do you want solution in flask labelled?(Y/N):	Y
Titrant increment (in mL) :	5.0

INSTRUCTIONS - TITRATION OF ACIDS AND BASES USING INDICATORS

Step 1: From Main Menu select option "2" .

MAIN MENU

Titration of acids & bases	**.......(1)**
Titration of acids & bases using indicators	**.......(2)**
Titration Quiz	**.......(3)**
Titration Assignment	**.......(4)**
Quit	**.......(Q)**

Select option....................(2)

Step 2: Same as for Titration of Acids and Bases.

Step 3: Same as for Titration of Acids and Bases.

Step 4: Select an indicator from the following:

INDICATORS AVAILABLE

Methyl orange (3.1-4.4)	**......(1)**
Methyl red (4.2-6.2)	**......(2)**
Litmus (4.5-8.3)	**......(3)**
Bromothymol blue (6.0-7.6)	**......(4)**
Phenolphthalein (8.3-10.0)	**......(5)**
Thymolphthalein (9.3-10.6)	**......(6)**
Ideal (equiv. pt.)	**......(7)**

Select option................................( )

Step 5: Same as step 4 for Titration of Acids and Bases.

INSTRUCTIONS - QUIZ

Step 1: From Main Menu select option "3" .

MAIN MENU	
Titration of acids & bases	**........(1)**
Titration of acids & bases using indicators	**........(2)**
Titration Quiz	**........(3)**
Titration Assignment	**........(4)**
Quit	**........(Q)**
Select option....................	**(3)**

Step 2: Program allows selection of unknown solution.

Test Solutions Available	
Hydrochloric acid	**.........(1)**
Sodium hydroxide	**.........(2)**
Acetic acid	**.........(3)**
Ammonia	**.........(4)**
Oxalic acid	**.........(5)**
Quit	**.........(Q)**
Select Solution...............	**( )**

The computer randomly selects the concentration of this solution to be in the range 0.100M to 1.000M. Titrations must carried out in order to determine this concentration.

Step 3: It is necessary to select concentration of the standard solution. The student is given the option of either selecting the concentration or allowing the computer to select the most suitable concentration of the standard solution. It is easier to allow the computer to determine the concentration, however in the ASSIGNMENT program the student is not given this option and must select the concentration himself. The Quiz program, will offer the student the appropriate skills required in selecting the concentration of the standard solution to accomplish the ASSIGNMENT program.

INSTRUCTIONS - ASSIGNMENT

The student must enter an assignment number between 0 and 99. This number corresponds to concentration according to the following table:

No.	Conc.(M)	No.	Conc.(M)	No.	Conc.(M)	No.	Conc.(M)
0	0.100	30	0.109	60	0.118	90	0.127
1	0.196	31	0.205	61	0.214	91	0.223
2	0.292	32	0.301	62	0.310	92	0.319
3	0.388	33	0.397	63	0.406	93	0.415
4	0.484	34	0.493	64	0.502	94	0.511
5	0.580	35	0.589	65	0.598	95	0.607
6	0.676	36	0.685	66	0.694	96	0.703
7	0.772	37	0.781	67	0.790	97	0.799
8	0.868	38	0.877	68	0.886	98	0.895
9	0.964	39	0.973	69	0.982	99	0.991
10	0.103	40	0.112	70	0.121		
11	0.199	41	0.208	71	0.217		
12	0.295	42	0.304	72	0.313		
13	0.391	43	0.400	73	0.409		
14	0.487	44	0.496	74	0.505		
15	0.583	45	0.592	75	0.601		
16	0.679	46	0.688	76	0.697		
17	0.775	47	0.784	77	0.793		
18	0.871	48	0.880	78	0.889		
19	0.967	49	0.976	79	0.985		
20	0.106	50	0.115	80	0.124		
21	0.202	51	0.211	81	0.220		
22	0.298	52	0.307	82	0.226		
23	0.394	53	0.403	83	0.412		
24	0.490	54	0.499	84	0.508		
25	0.586	55	0.595	85	0.604		
26	0.682	56	0.691	86	0.700		
27	0.778	57	0.787	87	0.796		
28	0.874	58	0.883	88	0.892		
29	0.970	59	0.979	89	0.988		

Assignment numbers 0 to 19 correspond to hydrochloric acid.
" " 20 to 39 " " sodium hydroxide.
" " 40 to 59 " " acetic acid.
" " 60 to 79 " " ammonia
" " 80 to 99 " " oxalic acid.

WORKSHEET 1: STRONG ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 1.1
Titrate 20 mL 1.000M HCl in flask with 1.000M NaOH in burette.

Vol. base	0.0	10 .0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.2
Titrate 20 mL 1.000M HNO3 in flask with 1.000M NaOH in burette.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.3
Titrate 20 mL 1.000M HClO4 in flask with 1.000M NaOH in burette.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.4
Titrate 20 mL 1.000M HCl in flask with 1.000M KOH in burette.

Vol. base	0.0	10 .0	19.0	20 .0	21.0	30.0	40.0
pH							

GRAPH: For each of the above titrations graph pH vs. volume of base.

QUESTIONS:
(a) How does the nature of the above acids (hydrochloric, nitric & perchloric) effect the pH titration curve?
(b) How does the nature of the above bases (sodium hydroxide & potassiumhydroxide) effect the pH titration curve?

WORKSHEET 2: STRONG ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 2.1
Titrate 20 mL 1.000M HCl in flask with 1.000M NaOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							

Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 2.2
Titrate 20 mL 0.100M HCl in flask with 0.100M KOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							

Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 2.3
Titrate 20 mL 0.010M HCl in flask with 0.010M KOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							

Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

GRAPH: For each of the above titrations plot pH vs. volume of base. Draw the three curves on the one graph.

QUESTION: How does the concentration of acid and base effect the pH titration curve?

WORKSHEET 3: STRONG ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 3.1

Titrate 20 mL 0.100M NaOH in flask with 0.100M HNO3 in burette.

Vol. acid	0.0	5 .0	15.0	19.0	19.5	20.0	20.5	30.0
pH								

Titration 3.2

Titrate 20 mL 0.100M NaOH in flask with 1.000M HNO3 in burette.

Vol. acid	0.0	1.0	2.0	3.0	4.0	5.0	6.0
pH							

Titration 3.3

Titrate 20 mL 0.100M NaOH in flask with 0.010M HNO3 in burette.

Vol. acid	0.0	10.0	20 .0	40.0	60.0	100.0	150.0
pH							

QUESTIONS:

(a) In any titration, why is it important that the concentration of acid and base are fairly close?

(b) Compare titration 3.1 with titration 2.2. Draw a sketch to illustrate how pH curve is affected by placing acid instead of base in burette.

WORKSHEET 4: WEAK ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 4.1

Titrate 20 mL 1.000M acetic acid in flask with 1.000M KOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							
Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 4.2

Titrate 20 mL 0.100M acetic acid in flask with 0.100M KOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							
Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 4.3

Titrate 20 mL 0.010M acetic acid in flask with 0.010M KOH in burette.

Vol. base	0.0	5.0	10.0	15.0	19.0	19.5	19.9
pH							
Vol.base	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

GRAPH: For each of the above titrations plot pH vs. volume of base. Draw the three curves on the one graph.

QUESTIONS:

(a) How does the concentration of acetic acid effect the pH titration curve?

(b) How does strength of acid effect the pH curve?

Compare titration 4.1 with titration 2.1;
" " 4.2 " " 2.2;
" " 4.3 " " 2.3.

WORKSHEET 5: WEAK ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 5.1

Titrate 50 mL 1.000M acetic acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10 .0	20.0	30 .0	40.0	49.0	49.9
pH							
Vol.base	50.0	50.1	51.0	60.0	70.0	80.0	100.0
pH							

Titration 5.2

Titrate 50 mL 1.000M hydrofluoric acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10 .0	20.0	30 .0	40.0	49.0	49.9
pH							
Vol.base	50.0	50.1	51.0	60.0	70.0	80.0	100.0
pH							

Titration 5.3

Titrate 50 mL 1.000M hydrocyanic acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10 .0	20.0	30 .0	40.0	49.0	49.9
pH							
Vol.base	50.0	50.1	51.0	60.0	70.0	80.0	100.0
pH							

GRAPH: For each of the above titrations plot pH vs. volume of base. Draw the three curves on the one graph.

QUESTIONS:

(a) Compare the strengths of the three different acids used in these titrations.

(b) How does strength of acid effect the titration? Compare above titrations.

WORKSHEET 6: WEAK BASE / STRONG ACID TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 6.1

Titrate 20 mL 1.000M ammonia in flask is with 1.000M HCl in burette.

Vol. acid	0.0	5 .0	10.0	15 .0	19.0	19.5	19.9
pH							
Vol.acid	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 6.2

Titrate 20 mL 0.100M ammonia in flask with 1.000M HCl in burette.

Vol. acid	0.0	5 .0	10.0	15 .0	19.0	19.5	19.9
pH							
Vol.acid	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

Titration 6.3

Titrate 20 mL 0.010M ammonia in flask with 1.000M HCl in burette.

Vol. acid	0.0	5 .0	10.0	15 .0	19.0	19.5	19.9
pH							
Vol.acid	20.0	20.1	20.5	21.0	25.0	30.0	40.0
pH							

GRAPH: For each of the above titrations plot pH vs. volume of acid. Draw the three curves on the one graph.

QUESTIONS:

(a) How does the concentration of ammonia effect the pH titration curve?

(b) How does strength of base effect the pH curve?

Compare titration 6.1 with titration 2.1;

" " 6.2 " " 3.2.

WORKSHEET 7: DIPROTIC ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 7.1

Titrate 20 mL 1.000M tartaric acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 7.2

Titrate 20 mL 1.000M oxalic acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 7.3

Titrate 20 mL 1.000M sulphuric acid in flask with 1.000M NaOH in burette.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

GRAPH: For each of the above titrations plot pH vs. volume of base. Draw the three curves on the one graph.

QUESTIONS:

(a) Compare the pH titration curves obtained for the three different diprotic acids.

(b) For these diprotic acids why is it easier to titrate to the second end point rather than to the first end point?

WORKSHEET 8: INDICATORS

Ideally, a suitable indicator will be a certain colour 0.1 mL before equivalence point and a different colour 0.1 mL after equivalence point.

Complete the following tables by indicating the colour of the solution at certain points around the equivalence point.

Titration 8.1
Titrate 25.0 mL 1.00M hydrochloric acid with 1.00M sodium hydroxide.

	Volume of NaOH (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

Titration 8.2
Titrate 25.0 mL 0.01M hydrochloric acid with 0.01M sodium hydroxide.

	Volume of NaOH (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

Titration 8.3
Titrate 25.0 mL 1.00M acetic acid with 1.00M sodium hydroxide.

	Volume of NaOH (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

QUESTION: Compare the above titrations and discuss the influence of concentration and strength of solutions on selection of indicator.

WORKSHEET 9: INDICATORS

Carry out the following simulated titrations and determine which of the available indicators would be suitable for the titration. It will be necessary to repeat each titration several times, each time selecting a different indicator.
Before commencing each titration calculate the equivalence point.

Titration 9.1
Titrate 25.0mL 0.15M perchloric acid with 0.10M potassium hydroxide.

Calculated equivalence point:

Suitable indicator(s):

Titration 9.2
Titrate 50.0 mL 1.00M sodium hydroxide with 0.80M acetic acid.

Calculated equivalence point:

Suitable indicator(s):

Titration 9.3
Titrate 50.0 mL 0.008M sodium hydroxide with 0.010M acetic acid.

Calculated equivalence point:

Suitable indicator(s):

Titration 9.4
Titrate 25.0 mL 0.012M ammonia with 0.010M hydrochloric acid.

Calculated equivalence point:

Suitable indicator(s):

Titration 9.5
Titrate 25.0 mL 1.000M tartaric acid with 0.900M sodium hydroxide.

Calculated equivalence point:

Suitable indicator(s):

Titration 9.6
Titrate 20.0 mL 1.000M oxalic acid in flask is with 0.850M potassium hydroxide.

Calculated equivalence point:

Suitable indicator(s):

WORKSHEET 10: ASSIGNMENT NO. ____

The most accurate results are obtained when the concentration of the standard solution is close to that of the unknown.

The concentration of the unknown should be determined to THREE significant figures. Only show data from your final or "best" titration.

Unknown solution:
(eg. acetic acid)

Data:

Molarity of standard solution =
Volume of standard solution =
Volume of unknown solution =

Calculations:

Concentration of unknown:

WORKSHEET 1: SALT OF WEAK ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 1.1

Titrate 20 mL 1.000M NaCN in flask with 1.000M HCl in burette, using ideal indicator.

Vol. base	0.0	10 .0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.2

Titrate 20 mL 1.000M CH_3COONa in flask with 1.000M HCl in burette, using ideal indicator.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.3

Titrate 20 mL 1.000M HCN in flask with 1.000M NaOH in burette using ideal indicator.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 1.4

Titrate 20 mL 1.000M CH_3COOH in flask with 1.000M KOH in burette using ideal indicator.

Vol. base	0.0	10 .0	19.0	20 .0	21.0	30.0	40.0
pH							

GRAPH: For each of the above titrations graph pH vs. volume titrant.

QUESTIONS:

(a) How does the strength of the weak acid from which the salt is derived effect the pH titration curve?

(b) Compare the acidity of the salt involved in titrations 1.1 and 1.2 with the strength of the weak acids in titrations 1.3 and 1.4.

WORKSHEET 2: SALT OF WEAK ACID / STRONG BASE TITRATIONS

Ideally, a suitable indicator will be a certain colour 0.1mL before equivalence point and a different colour 0.1mL after equivalence point.

Complete the following tables by indicating the colour of the solution at certain points around the equivalence point.

Titration 2.1

Titrate 25.0 mL 1.00M $NaCH_3COO$ with 1.00M HCl.

	Volume of HCl (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

Titration 2.2

Titrate 25.0 mL 0.01M $NaCH_3COO$ with 1.00M HCl.

	Volume of HCl (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

Titration 2.3

Titrate 25.0 mL 1.00M HCN with 1.00M HCl.

	Volume of HCl (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) bromothymol blue			
(c) phenolphthalein			

QUESTION: Compare the above titrations and discuss the influence of concentration and strength of solutions on selection of indicator.

WORKSHEET 3: SALT OF STRONG ACID / WEAK BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 3.1
Titrate 20 mL 1.000M NH_4Cl in flask with 1.000M NaOH in burette, using ideal indicator.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 3.2
Titrate 20 mL 1.000M NH_3 in flask with 1.000M HCl in burette, using ideal indicator.

Vol. base	0.0	10.0	19.0	20.0	21.0	30.0	40.0
pH							

Titration 3.3
Titrate 25.0 mL 1.00M NH_4Cl with 1.00M HCl.

	Volume of HCl (mL)		
Indicator	24.90	25.00	25.10
(a) methyl orange			
(b) methyl red			
(c) bromothymol blue			
(d) thymolphthalein			
(e) phenolphthalein			

GRAPH: For each of the above titrations graph pH vs. volume titrant.

QUESTIONS:
(a) How does the strength of the weak acid from which the salt is derived effect the pH titration curve?
(b) Compare the acidity of the salt involved in titration 3.1 with the strength of the weak base in titration 3.3.

WORKSHEET 4: SALTS OF DIPROTIC ACID / STRONG BASE TITRATIONS

Carry out the following simulated titrations and complete the tables.

Titration 4.1

Titrate 20 mL 1.000M sodium carbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 4.2

Titrate 20 mL 1.000M potassium phthalate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 4.3

Carry out a series of titrations using 20 mL 1.000M sodium carbonate in flask and 1.000M HCl in burette. In each titration use a different indicator. Determine which of the following indicators is best suited to determining the first end point, and which is best suited to determining the second end point:

(a) methy orange;
(b) methyl red;
(c) litmus;
(d) bromothymol blue;
(e) thymolphthalein.

GRAPH: For each of the titrations 4.1 and 4.2 plot pH vs. volume of base. Draw the two curves on the one graph.

QUESTIONS:

(a) Compare the pH titration curves obtained for the two different diprotic salts.

(b) Compare the acidity of the diprotic salt with the strength of the corresponding diprotic acids from which the salts are derived.

WORKSHEET 5: MIXTURE OF SALTS

Carry out the following simulated titrations and complete the tables.

Titration 5.1
Titrate 20 mL 1.000M sodium carbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 5.2
Titrate 20 mL 1.000M sodium bicarbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 5.3
Titrate 20 mL of a mixture 1.000M sodium bicarbonate and 1.000M sodium bicarbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							
Vol.base	55.0	60.0	65.0	70.0	75.0	80.0	90.0
pH							

QUESTION: Compare the end points of the titration involving a mixture of salts (titration 5.3) with the end points of two separate titrations (titrations 5.1 &5.2)

WORKSHEET 6: MIXTURE OF SALTS

Carry out the following simulated titrations and complete the tables.

Titration 6.1

Titrate 20 mL of a mixture 0.500M sodium bicarbonate and 0.500M sodium bicarbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							

Titration 6.2

Titrate 20 mL of a mixture 1.000M sodium bicarbonate and 0.500M sodium bicarbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							
Vol.base	55.0	60.0	65.0	70.0	75.0	80.0	90.0
pH							

Titration 6.3

Titrate 20 mL of a mixture 0.500M sodium bicarbonate and 1.500M sodium bicarbonate in flask with 1.000M HCl in burette using ideal indicator.

Vol. base	0.0	10.0	15 .0	19.0	20.0	21.0	25.0
pH							
Vol.base	30.0	35.0	39.0	40.0	41.0	45.0	50.0
pH							
Vol.base	55.0	60.0	65.0	70.0	75.0	80.0	90.0
pH							

QUESTION: Compare the end points of the above titrations involving a mixture of salts with the end points of two separate titrations (titrations 5.1 &5.2)

WORKSHEET 7: ASSIGNMENT NO. ____

The most accurate results are obtained when the concentration of the standard solution is close to that of the unknown.

The concentration of the unknown should be determined to THREE significant figures. Only show data from your final or "best" titration.

Unknown solution:
(eg. sodium acetate solution)

Data:

Molarity of standard solution =
Volume of standard solution =
Volume of unknown solution =

Calculations:

Concentration of unknown:

WORKSHEET 1A: REACTION BETWEEN AN ACTIVE METAL AND AN ACID

MACROSCOPIC REACTION

Question 1. What gas is produced when magnesium is reacted with hydrochloric acid?

____

Question 2. How is this gas identified?

____

Question 3. Compare the reaction between magnesium and hydrochloric acid, with the reaction between nickel and hydrochloric acid.

Similarities: ______________________________

____

Differences: ______________________________

Question 4. Write the general word equation for a reaction between an active metal and an acid.

____

Question 5. Write balanced chemical equations for the reactions of magnesium and nickel with hydrochloric acid.

____

____

Question 6. Name three reactive metals (not including magnesium and nickel) : ______________________________

Refer to your textbook to answer the following question.

Question 7. From what source is magnesium metal obtained?

____

WORKSHEET 1B: REACTION BETWEEN AN ACTIVE METAL AND AN ACID

MICROSCOPIC REACTION

Question 8. What is a hydronium ion? ____________________

____

Question 9. What is the charge on a magnesium ion? ________

Question 10. What happens to the polarity of the hydronium ion as it approaches the metal atoms? ____________________

____

Question 11. During the reaction between magnesium and hydronium ions , how many electrons are lost by each magnesium atom? ____________

Question 12. During the reaction between calcium and hydronium ions, how many electrons are lost by each calcium atom? ______

Question 13. What happens to the electron(s) lost by the metal?

____

____

Question 14. Write the net (half) reaction between hydronium ions and electrons. ____________________

Question 15. What part does the chloride ion (from the hydrochloric acid) play in the reaction? ____________

____

Question 16. Why do you think nickel reacts more slowly than magnesium with hydrochloric acid. ____________________

____

WORKSHEET 2A: REACTION BETWEEN A VERY ACTIVE METAL AND WATER

MACROSCOPIC REACTION

Question 1. What gas is produced when sodium is placed into a beaker of water?

____

Question 2. How is this gas identified?

____

Question 3. Comment on the acidity of the solution produced from the reaction of sodium with water.

____

Question 4. Compare the reaction of calcium and water, with the reaction of sodium and water.

Similarities: ______________________________

____

Differences: ______________________________

Question 5. Write the general word equation for a reaction between a very active metal and water.

____

Question 6. Write balanced chemical equations for the reactions of sodium with water and calcium with water.

____

____

Refer to your textbook to answer the following questions.

Question 7. Name three very reactive metals (not including sodium and calcium) : __________________________

Question 8. From what source is sodium metal obtained?

____

MICROSCOPIC REACTION

Question 9. What is the charge on a sodium ion? ____________

Question 10. What is the charge on a calcium ion? ____________

Question 11. Describe what happens to the polarity of a water molecule as it approaches the surface of a metal?

____

Question 12. During the reaction between sodium and water, how many electrons are lost by each sodium atom? ____________

Question 13. During the reaction between calcium and water, how many electrons are lost by each calcium atom? ____________

Question 14. Comment on the relative sizes of the metal atom and the corresponding metal ions. ______________________

____

Question 15. What happens to the electron(s) lost by the metal?

____

____

Question 16. Why does the solution become basic during this reaction. ____

Question 17. Write the net (half) reaction between water and electrons. ____

Question 18. What is a hydrated metal ion? ____________

____

WORKSHEET 3A: REACTION BETWEEN A CARBONATE AND AN ACID

MACROSCOPIC REACTION

Question 1. What gas is produced when sodium carbonate is reacted with hydrochloric acid?

____

Question 2. How is this gas identified?

____

Question 3. Write the general word equation for a reaction between a carbonate and an acid.

____

Question 4. Write balanced chemical equations for the reactions of sodium carbonate and magnesium carbonate with hydrochloric acid.

____

____

Question 5. Write balanced chemical equations for the reactions of sodium carbonate and magnesium carbonate with sulfuric acid.

____

____

Question 6. Write balanced chemical equations for the reactions of calcium carbonate and lead carbonate with nitric acid.

____

____

MICROSCOPIC REACTION

Question 7. What is the charge on the carbonate ion? ________

Question 8. What ion is formed by the reaction of one hydronium ion with a carbonate ion? ________________________

Question 9. What species is formed by the reaction of two hydronium ions with a carbonate ion? ________________

Question 10. What role does the hydronium ion play in this reaction? ____

____

Question 11. What happens to the metal ions in these reactions?

____

____

Question 12. Calcium carbonate and sodium carbonate both react with hydrochloric acid. Compare these two reactions.

Similarities ______________________________________

____

Differences ______________________________________

____

Question 13. What part does the chloride ion (from the hydrochloric acid) play in the reaction? ______________

____

WORKSHEET 4: REACTIONS OF LITMUS

PART A: MACROSCOPIC REACTION

Question 1. What happens when red and blue litmus paper are dipped into an acidic solution? ______________________

____

Question 2. What happens when red and blue litmus paper are dipped into a basic solution? ______________________

____

Question 3. What is the difference between using litmus paper and litmus solution in detecting the acidity of a solution?

____

Refer to your textbook to answer the following question:
Question 4. Name two other indicators (other than litmus) that are commonly used to determine the acidity of a solution:

____

PART B: MICROSCOPIC REACTION

Question 5. What is the difference between red litmus and blue litmus molecules? ______________________

Question 6. What happens when a hydrogen ion is added to blue red litmus molecules? ______________________

Question 7. What happens when a hydrogen ion is removed from red litmus molecules? ______________________

Question 8. Describe the reaction that occurs when blue litmus is placed into an acidic solution. ______________________

____

Question 9. Describe the reaction that occurs when red litmus is placed into basic solution. ______________________

____

```
(*$S++,I-,R-*)
UNIT USEFUL; INTRINSIC CODE 25 DATA 26;

INTERFACE

USES TURTLEGRAPHICS;
CONST
  SPACE=' ';
  XMIN=0; YMIN=0; XMAX=279; YMAX=191;         (*dimensions of graphics screen*)
TYPE
  CHARSET=SET OF CHAR;
  SHORTSTR=STRING[8];
  BYTE=0..255;
  MEMLOC = PACKED ARRAY [0..1] OF byte;
  ACCESS = RECORD
             CASE BOOLEAN OF
                TRUE : (ADDRESS:INTEGER);
                FALSE: (POINTER:^MEMLOC);
             END;
VAR  RET: CHAR;

PROCEDURE BEEP;
PROCEDURE SHIFTUP(VAR CH:CHAR);
PROCEDURE GETACHAR(VAR CH:CHAR; LEGALSET:CHARSET);
PROCEDURE GETRESPONSE(X,Y:INTEGER; VAR S:SHORTSTR; MAXLEN: INTEGER;
                                                LEGALSET:CHARSET);
PROCEDURE GETTEXTCHAR(X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET);
PROCEDURE GETHIRESPONSE(INITX,Y:INTEGER; VAR S:SHORTSTR; MAXLEN:INTEGER;
                                                LEGALSET:CHARSET);
PROCEDURE REMOVERESPONSE(X,Y,LEN:INTEGER);
PROCEDURE GETHICHAR(X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET);
PROCEDURE WSTAT(X,Y:INTEGER; S:STRING);
PROCEDURE FILLBOX(LEFT,RIGHT,BOTTOM,TOP: INTEGER; COLOUR:SCREENCOLOR);
PROCEDURE MOVECOL(X,Y:INTEGER; COL:SCREENCOLOR);
PROCEDURE DRAWLINE(X1,Y1,X2,Y2:INTEGER; COL: SCREENCOLOR);
FUNCTION AROW(NUM:INTEGER; CH:CHAR):CHAR;
FUNCTION AT(X,Y:INTEGER):CHAR;
PROCEDURE WAIT(TIME:INTEGER);
FUNCTION PEEK(ADDRS: INTEGER):BYTE;
FUNCTION KEYIN:BOOLEAN;

IMPLEMENTATION

(*************************************************************)
PROCEDURE BEEP;
(*************************************************************)
BEGIN
  WRITE(CHR(7));
END;

(*************************************************************)
PROCEDURE SHIFTUP(*VAR CH:CHAR*);
(*************************************************************)
BEGIN
   IF CH IN [CHR(97)..CHR(122)] THEN  CH:=CHR(ORD(CH)-32)
END; (*SHIFTUP*)
```

```
(****************************************************************)
PROCEDURE GETACHAR(*VAR CH: CHAR; LEGALSET:CHARSET*);
(****************************************************************)
BEGIN
 REPEAT
     READ(KEYBOARD,CH);
     IF EOLN(KEYBOARD) THEN CH:=RET;
     SHIFTUP(CH);
 UNTIL CH IN LEGALSET;
END;  (* GETACHAR *)

(****************************************************************)
PROCEDURE GETRESPONSE(*X,Y:INTEGER; VAR S: SHORTSTR; MAXLEN: INTEGER;
                                                  LEGALSET:CHARSET*);
(****************************************************************)
(* GETS A STRING 'S' OF MAXLENGTH FROM TEXT SCREEN.
   ONLY CHARACTERS IN LEGALSET WILL BE ACCEPTED & ECHOED TO SCREEN*)
VAR CH,BACK: CHAR;
    COMMAND: CHARSET;
    COMPLETE:BOOLEAN;
    S1:STRING[1];
BEGIN
   BACK:=CHR(8);
   COMMAND:=[RET,BACK];
   S:='';
   S1:=' ';
   GOTOXY(X,Y);
  REPEAT
     GETACHAR(CH,LEGALSET+COMMAND);
     COMPLETE:=CH=RET;
    IF CH IN LEGALSET THEN
     BEGIN
       IF LENGTH(S)<MAXLEN THEN
        BEGIN
          S1[1]:=CH;
           S:=CONCAT(S,S1);
          WRITE(CH);
        END;
     END
      ELSE IF ((CH=BACK) AND (LENGTH(S)>0)) THEN
       BEGIN
          GOTOXY(X+LENGTH(S)-1,Y);
          WRITE(' ');
          GOTOXY(X+LENGTH(S)-1,Y);
          DELETE(S,LENGTH(S),1);
       END;
   UNTIL ((COMPLETE) AND (LENGTH(S)>0));
 END;  (*GETRESPONSE *)
```

```
(****************************************************************)
PROCEDURE GETTEXTCHAR(*X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET*);
(****************************************************************)
(* GETS A SINGLE CHARACTER FROM TEXT SCREEN - RETURN REQUIRED.
  ONLY CHARACTERS IN LEGALSET WILL BE ACCEPTED & DISPLAYED ON SCREEN*)
BEGIN
    GETRESPONSE(X,Y,S,1,LEGALSET);
    ACH:=S[1];
END;  (* GETTEXTCHAR*)

(****************************************************************)
PROCEDURE GETHIRESPONSE(*INITX,Y : INTEGER; VAR S: SHORTSTR;
                    MAXLEN: INTEGER; LEGALSET:CHARSET*);
(****************************************************************)
(* GETS A STRING 'S' FROM HI RES SCREEN OF MAXLENGTH.
  ONLY CHARACTERS IN LEGALSET WILL BE ACCEPTED & ECHOED TO SCREEN*)
VAR   X:INTEGER;  CH,                    (* INPUT CHAR *)
      BACK: CHAR;                        (* BACKSPACE *)
      COMMAND:  CHARSET;
      COMPLETE  : BOOLEAN;
      S1 : STRING[1];
BEGIN
  BACK:=CHR(8);            (* BACKSPACE *)
   COMMAND:=[RET,BACK];
   S:='';
   S1:=' ';
   X:=INITX;
   MOVETO(X,Y);
  REPEAT
     GETACHAR(CH,LEGALSET+COMMAND);
     COMPLETE:=CH=RET;
    IF CH IN LEGALSET THEN
     BEGIN
       IF LENGTH(S)<MAXLEN THEN
        BEGIN
          S1[1]:=CH;
           S:=CONCAT(S,S1);
          WCHAR(CH);
        END
        ELSE BEEP
      END

      ELSE IF ((CH=BACK) AND (LENGTH(S)>0)) THEN
       BEGIN
          X:=TURTLEX-7;
         IF X>=INITX THEN
          BEGIN
             DELETE(S,LENGTH(S),1);
             MOVETO(X,Y);
             WCHAR(' ');
             MOVETO(X,Y);
          END;
       END;
  UNTIL ((COMPLETE) AND (LENGTH(S)>0));
END; (*GETHIRESPONSE*)
```

```
(**********************************************************)
PROCEDURE GETHICHAR(*X,Y:INTEGER; VAR ACH:CHAR; LEGALSET:CHARSET*);
(**********************************************************)
(* GETS A SINGLE CHARACTER FROM HIRES SCREEN - RETURN REQUIRED.
  ONLY CHARACTERS IN LEGALSET WILL BE ACCEPTED & DISPLAYED ON SCREEN*)
BEGIN
    GETHIRESPONSE(X,Y,S,1,LEGALSET);
   ACH:=S[1];
END;  (* GETHICHAR*)

(**********************************************************)
PROCEDURE REMOVERESPONSE(*X,Y,LEN: INTEGER*);
(**********************************************************)
VAR K : INTEGER;
BEGIN
   MOVETO(X,Y);
   FOR K:=1 TO LEN DO WCHAR(' ');
END;

(**********************************************************)
PROCEDURE WSTAT(*X,Y:INTEGER; S:STRING*);
(**********************************************************)
BEGIN
   MOVETO(X,Y);
   WSTRING(S);
END;

(**********************************************************)
PROCEDURE  FILLBOX(*LEFT,RIGHT,BOTTOM,TOP:INTEGER;COLOUR:SCREENCOLOR*);
(**********************************************************)
BEGIN
    VIEWPORT(LEFT,RIGHT,BOTTOM,TOP);
   FILLSCREEN(COLOUR);
    VIEWPORT(XMIN,XMAX,YMIN,YMAX);
END;

(**********************************************************)
PROCEDURE MOVECOL(*X,Y:INTEGER; COL:SCREENCOLOR*);
(**********************************************************)
BEGIN
   MOVETO(X,Y);
   PENCOLOR(COL);
END;

(**********************************************************)
PROCEDURE DRAWLINE(*X1,Y1,X2,Y2: INTEGER; COL:SCREENCOLOR*);
(**********************************************************)
BEGIN
  MOVECOL(X1,Y1,COL);
  MOVECOL(X2,Y2,NONE);
END;
```

```
(*************************************************************)
 FUNCTION AROW(*NUM:INTEGER;  CH:CHAR):CHAR*);
(*************************************************************)
 VAR J:INTEGER;
 BEGIN
    FOR J:=1 TO NUM DO WRITE(CH);
     AROW:=CHR(0);
 END; (* AROW *)

(*************************************************************)
FUNCTION  AT(*X,Y:INTEGER):CHAR*);
(*************************************************************)
BEGIN
   GOTOXY(X,Y);
    AT:=CHR(0);
END; (* AT *)

(*************************************************************)
PROCEDURE WAIT(*TIME: INTEGER*);
(*************************************************************)
VAR J,DELAY:INTEGER;
BEGIN
    FOR DELAY:=1 TO TIME DO  J:=DELAY*DELAY*DELAY;
END; (* WAIT *)

(*************************************************************)
FUNCTION  PEEK(*ADDRS:INTEGER):BYTE*);
(*************************************************************)
VAR MEMORY:ACCESS;
BEGIN
   MEMORY.ADDRESS:=ADDRS;
   PEEK:=MEMORY.POINTER^[0];
END;

(*************************************************************)
FUNCTION KEYIN(*:BOOLEAN*);
(*************************************************************)
CONST  KEYBD= -16384;
BEGIN
    IF PEEK(KEYBD)>128 THEN KEYIN:=TRUE  ELSE KEYIN:=FALSE;
END;


BEGIN
  RET:=CHR(13);
END.
```

```
(*$S++,R-,V-*)
(* Library unit used by four titration programs - TITRATION, INDICATOR, ASSIGNMENT,
QUIZ  and INTRO (Startup for Titration package) *)

UNIT TITRLIB;  INTRINSIC CODE 18 DATA 19;

INTERFACE

USES  TURTLEGRAPHICS,TRANSCEND,USEFUL;

CONST
   FLASKX=60; FLASKY=24;                       (*co-ordinates of centre base of flask*)
   FLASKSIZ=100;                                                    (*size of flask*)
   KW=1.0E-14;                                         (*ionization constant of water*)

TYPE
   ACIDORBASE=(ACID,BASE);
    TITRAT=(WEAKACID,STRONGACID,WEAKBASE,DIPROTIC);

VAR
   TITRTYPE: TITRAT;                                     (*type of current titration*)
   FLASKTOP,                                                (*base of neck of flask*)
   NECKTOP: INTEGER;                                     (* very top of neck of flask *)
   K1,K2,                                                 (*ionization constants *)
   FLASKVOL,                                               (*vol. of soln in flask*)
   HCONC,OHCONC:REAL;                                    (*conc. of acid & base *)
   FILLRATE:INTEGER;                          (*determines rate at which flask filled*)
   INFLASK: ACIDORBASE;                                      (*type of soln in flask*)
   QUIT: BOOLEAN;
   NUMS:CHARSET;


FUNCTION RVALUE(VAR S:STRING):REAL;
PROCEDURE REALSTR(VAR REALNUM:REAL; VAR WORD:STRING; DECPOINT,SIZE:INTEGER);
PROCEDURE INRANGERESPONSE(VAR VALUE:REAL; VAR S:SHORTSTR;
                                                MIN,MAX:REAL;   X,Y: INTEGER);
PROCEDURE  DOUBLELINE(X1,Y1,X2,Y2:INTEGER;  COL:SCREENCOLOR);
PROCEDURE  DRAWBOX(X1,Y1,X2,Y2:INTEGER;  COL:SCREENCOLOR);
PROCEDURE INITSCREEN;
PROCEDURE  DRAWAXES(X,Y,SIZE:  INTEGER;  COL:SCREENCOLOR);
PROCEDURE  DRAWFLASK(X,Y,SIZE:INTEGER;  COL:SCREENCOLOR);
PROCEDURE FILLFLASK(VAR LTSIDE,RTSIDE,OLDLEVEL,INCREASE:INTEGER;
PROCEDURE MOVEDROP(INCR: REAL; VAR LIQLEVEL:INTEGER);
PROCEDURE  ACIDMOLARITY(S:SHORTSTR);
PROCEDURE  BASEMOLARITY(S:SHORTSTR);
PROCEDURE  ACIDISP(S:SHORTSTR);
PROCEDURE  BASEDISP(S:SHORTSTR);
PROCEDURE  DISPLAYPH(S:SHORTSTR);
PROCEDURE  TWOPROMPTS(S1,S2:  STRING);
PROCEDURE SETUPCONDITIONS(VAR HCONC,OHCONC: REAL;
                 VAR INFLASK: ACIDORBASE);
PROCEDURE CHECKKEY(var spacepr,selectchange: boolean);
PROCEDURE REQUEST;
PROCEDURE  INCRPROMPT;
PROCEDURE SELECTINR(VAR INCR: REAL);
PROCEDURE CHANGEINC(VAR CHANG:BOOLEAN; VAR INCR: REAL);
```

```
            COL:SCREENCOLOR);
PROCEDURE CLEARVALUES(VAR NEWCONC:BOOLEAN);
PROCEDURE SETCOLOUR;
PROCEDURE BACKTOMENU;
PROCEDURE GETK(VAR K1,K2 :REAL);
PROCEDURE SELECTTYPE(VAR TITRTYPE: TITRAT);
PROCEDURE CALCPH(ANYACID,ANYBASE,HCONC,OHCONC: REAL; VAR PH: REAL);

IMPLEMENTATION

(*****************************************************************)
FUNCTION RVALUE(*VAR S:STRING):REAL;*);
(*****************************************************************)
VAR  I,Z: INTEGER;
     A,F : REAL;
     CH:CHAR;  NEG:BOOLEAN;  GS:STRING;
BEGIN
  I:=1;
   Z:=ORD('0');
  A:=0;
   GS:=CONCAT(S,'/');
  WHILE GS[1]=SPACE DO DELETE(GS,1,1);
  CH:=GS[I];
  IF CH='-' THEN
   BEGIN
      NEG:=TRUE;
      I:=I+1;
     CH:=GS[I]
   END
   ELSE
    BEGIN
      NEG:=FALSE;
      IF CH='+' THEN
       BEGIN
         I:=I+1;
         CH:=GS[I]
       END
    END;
   IF CH IN ['0'..'9'] THEN
    REPEAT  (*GET INTEGER PART*)
      A:=10*A + ORD(CH)-Z;
      I:=I+1;
      CH:=GS[I]
     UNTIL NOT (CH IN ['0'..'9']);
     IF CH='.' THEN
      BEGIN
        F:=0.1;
        I:=I + 1;
        CH:=GS[I];
        WHILE CH IN ['0'..'9'] DO
         BEGIN
           A:=A + (ORD(CH)-Z)*F;
           F:=F*0.1;
           I:=I + 1;
           CH:=GS[I]
         END
```

```
     END;
    IF NEG THEN A:=-A;
    RVALUE:=A
  END;  (*RVALUE*)

(*****************************************************************)
PROCEDURE REALSTR(*VAR REALNUM:REAL; VAR WORD:STRING;
                                              DECPOINT,SIZE:INTEGER*);
(*****************************************************************)
(*Converts a real no. into a string displaying given no. decimal  pts. & string of certain
length, size.  N.B. This procedure is NOT suitable for all such conversions but is suitable for
the range of real no. in this program   *)
VAR   POWERTEN  :REAL;
       J,INTNUM  :INTEGER;
BEGIN
     REALNUM:=ABS(REALNUM);
    POWERTEN:=1.0;
    FOR J:=1 TO DECPOINT DO  POWERTEN:=POWERTEN*10;
    IF REALNUM<(32000.0/POWERTEN) THEN
     BEGIN
       INTNUM:=ROUND(REALNUM*POWERTEN);
      STR(INTNUM,WORD);
       IF LENGTH(WORD)>DECPOINT THEN  INSERT('.',WORD,LENGTH(WORD)-DECPOINT+1)
       ELSE
        BEGIN
            WHILE LENGTH(WORD)<DECPOINT DO  WORD:=CONCAT('0',WORD);
            WORD:=CONCAT('0.',WORD);
        END;
       IF RVALUE(WORD)<0.0 THEN WORD:=CONCAT('-',WORD);
      WHILE LENGTH(WORD)<SIZE DO WORD:=CONCAT(' ',WORD);
     END

     ELSE
       WORD:='NOT POSSIBLE';
                   (* Doesn't occur during this program*)
END;  (*  REALSRT *)

(*****************************************************************)
PROCEDURE INRANGERESPONSE(*VAR VALUE:REAL;  VAR S:SHORTSTR;
                                        MIN,MAX: REAL;   X,Y: INTEGER*);
(*****************************************************************)
(* Obtains a real no., VALUE, from hi-res screen in range MIN to MAX.VALUE is first
received as a string S and then converted to a real no. Response is erased from screen
when return pressed. 'Q' is accepted to quit the input.   *)
CONST  MAXLEN=5;
VAR   RANGE    : BOOLEAN;
BEGIN
 REPEAT
     GETHIRESPONSE(X,Y,S,MAXLEN,NUMS+['Q']);
    REMOVERESPONSE(X,Y,LENGTH(S));
    QUIT:=S='Q';
   IF NOT QUIT THEN
    BEGIN
      VALUE:=RVALUE(S);
       RANGE:=(MIN<=VALUE) AND (VALUE<=MAX);
       IF POS('Q',S)>1  THEN RANGE:=FALSE;
```

```
      IF (NOT RANGE) THEN BEEP;
    END;
  UNTIL RANGE OR QUIT;
END; (*INRANGERESPONSE*)

(*************************************************************************)
PROCEDURE DOUBLELINE(*X1,Y1,X2,Y2:INTEGER; COL: SCREENCOLOR*);
(*************************************************************************)
BEGIN
  DRAWLINE(X1,Y1,X2,Y2,COL);
  DRAWLINE(X1+1,Y1+1,X2+1,Y2+1,COL);
END;

(*************************************************************************)
PROCEDURE DRAWBOX(*X1,Y1,X2,Y2: INTEGER; COL: SCREENCOLOR*);
(*************************************************************************)
(* x1,y1 are coord. of bottom left hand corner & x2,y2 coord. of  top right hand corner of
box to be displayed *)
VAR  DOUBLE : INTEGER;
BEGIN
  FOR DOUBLE :=1 TO 2 DO
    BEGIN
      MOVECOL(x1,y1,COL);
      MOVETO(x2,y1);
      MOVETO(x2,y2);
      MOVETO(x1,y2);
      MOVECOL(x1,y1,NONE);
      X1:=X1+1;X2:=X2+1;Y1:=Y1+1;Y2:=Y2+1;
    END;
END;

(*************************************************************************)
PROCEDURE INITSCREEN;
(*************************************************************************)
(*draws burette with the lower end of burette at (x,y) and size being the length of the
burette*)

    (*=================================================================*)
     PROCEDURE DRAWBURETTE(X,Y,SIZE: INTEGER; COL : SCREENCOLOR);
    (*=================================================================*)
    VAR   WIDTH,LENGTH : INTEGER;

        (*---------------------------------------------------------------*)
        PROCEDURE ONESIDE(X1,Y1,WWIDTH,LENGTH : INTEGER);
        (*---------------------------------------------------------------*)
        BEGIN
          X1:=X1-WWIDTH;
          MOVECOL(X1,Y1,COL);
          Y1:=Y1+LENGTH;
          MOVETO(X1,Y1);
          Y1:=Y1+ABS(2*WWIDTH);
          X1:=X1-2*WWIDTH;
          MOVETO(X1,Y1);
          MOVECOL(X1,Y+SIZE,NONE);
        END;
```

```
   (*------------------------------------------------------------------*)
   PROCEDURE TAP;
   (*------------------------------------------------------------------*)
   BEGIN
    DRAWBOX(X-WIDTH,Y+LENGTH-6,X+WIDTH,Y+LENGTH-2,COL);
    FILLBOX(X-WIDTH+1,X+WIDTH-1,Y+LENGTH-5,Y+LENGTH-3,BLACK1);
   END;

BEGIN (* DRAWBURETTE*)
 WIDTH:=SIZE DIV 8;
 LENGTH:=SIZE DIV 3;
 ONESIDE(X,Y,WIDTH DIV 2,LENGTH);
 ONESIDE(X,Y,-WIDTH DIV 2,LENGTH);
 TAP;
END; (* DRAWBURETTE*)

(*====================================================================*)
PROCEDURE SETUPBOXES(COL : SCREENCOLOR);
(*====================================================================*)
(*if any box sizes or positions are altered then procedures such asACIDISP,
 ACIDMOLARITY etc. which display values in these boxes must also be altered *)
CONST LOWERY=170;
      TOPY=190;    (* y coord. of small boxes *)

   (*------------------------------------------------------------------*)
   PROCEDURE LABELS(X: INTEGER; CH1,CH2:SHORTSTR);
   (*------------------------------------------------------------------*)
   CONST HEIGHT=179; (*y coord of title of small boxes *)
   BEGIN
    WSTAT(X,HEIGHT,CH1);
    WSTAT(X+7,HEIGHT-7,CH2);
   END;

BEGIN (*SETUPBOXES*)
 DRAWBOX(1,20,135,165,COL);        (* main left box *)
 DRAWBOX(143,20,275,165,COL);      (* main right box*)
 DRAWBOX(1,LOWERY,64,TOPY,COL);    (* acid molarity *)
 LABELS(7,'M','a');
 DRAWBOX(70,LOWERY,135,TOPY,COL);  (* base molarity *)
 LABELS(78,'M','b');
 DRAWBOX(143,LOWERY,206,TOPY,COL); (* acid volume   *)
 LABELS(149,'V','a');
 DRAWBOX(212,LOWERY,275,TOPY,COL); (* base volume   *)
 LABELS(218,'V','b');
 DRAWBOX(84,135,134,164,COL);      (* set up pH box *)
END; (*SETUPBOXES*)
```

```
BEGIN  (* INITSCREEN *)
 INITTURTLE;
 TEXTMODE;
 SETUPBOXES(BLUE);  (*to hold molarity & vol of acid & base & pH*)
 DRAWBURETTE(FLASKX,FLASKY+FLASKSIZE,38,WHITE1);
      (* burette bottom  centre of tip  at x,y  of length  *)
END; (*  INITSCREEN*)

(*************************************************************)
PROCEDURE DRAWAXES(*X,Y,SIZE : INTEGER; COL :SCREENCOLOR*);
(*************************************************************)
(* ORIGIN OF GRAPH OF LENGTH SIZE *)
BEGIN
 WSTAT(X-10,Y-4+(SIZE DIV 2),'7-');
 WSTAT(X-8,Y+SIZE+2,'pH');        (* label y-axis*)
 WSTAT(X+SIZE-15,Y-9,'Vol');      (* label x-axis*)
 DRAWLINE(X,Y+SIZE,X,Y,COL);      (* y-axis of graph*)
 DRAWLINE(X,Y,X+SIZE,Y,COL);      (* x-axis of graph*)
END;

(*************************************************************)
PROCEDURE DRAWFLASK(*X,Y,SIZE : INTEGER; COL :SCREENCOLOR*);
(*************************************************************)
(* (x,y) is centre of flask of height and width of SIZE *)
VAR  RUN,RISE,WIDTH : INTEGER;

    (*==========================================================*)
    PROCEDURE ONESIDE(X1,Y1  : INTEGER);
    (*==========================================================*)
    BEGIN
     MOVECOL(X1,Y1,COL);
     MOVETO(X1+RUN,Y+RISE);
     MOVECOL(X1+RUN,Y+SIZE,NONE);
    END;

 BEGIN (*DRAWFLASK*)
  WIDTH:=SIZE DIV 2;
  RUN:=3*SIZE DIV 8;
  RISE:=2*RUN;
  ONESIDE(X-WIDTH,Y);
  RUN:=-RUN;
  ONESIDE(X+WIDTH,Y);
  DRAWLINE(X+WIDTH,Y,X-WIDTH,Y,COL);
 END; (*  DRAWFLASK *)

(*************************************************************)
PROCEDURE FILLFLASK(*VAR LTSIDE,RTSIDE,OLDLEVEL,INCREASE : INTEGER;
                   COL :SCREENCOLOR*);
(*************************************************************)
CONST  SLOPE=2;          (* Flask is drawn with sides of slope *)
VAR    Y,K ,
       NEWLEVEL : INTEGER;              (* Counter for 'FOR' loop *)
```

```
(*=================================================================*)
 PROCEDURE COMMENT;(*FLASK OVERFLOWS *)
 (*================================================================*)
 VAR S1,S2  : STRING; CH:CHAR;
 BEGIN
    CHARTYPE(6); REQUEST; CHARTYPE(10);  (* erase*)
    S1 :='Poor technique - your titration';
    S2:='has been terminated. Press <Q>';
    TWOPROMPTS(S1,S2);
    GETACHAR(CH,['Q']);
    CHARTYPE(6); TWOPROMPTS(S1,S2); CHARTYPE(10); (*erase*)
    QUIT :=TRUE;
 END; (* COMMENT *)

(*=================================================================*)
PROCEDURE FILL;
(*=================================================================*)
BEGIN
 IF COL=BLACK1 THEN DRAWLINE(LTSIDE,Y,RTSIDE,Y,WHITE1); (*display *)
 DRAWLINE(LTSIDE,Y,RTSIDE,Y,COL);
 Y:=Y+1;
END;

BEGIN (*FILLFLASK*)
 Y:=OLDLEVEL;
 NEWLEVEL:=OLDLEVEL+INCREASE;
 IF Y<FLASKTOP THEN
  BEGIN
   REPEAT
     LTSIDE:=LTSIDE+1;
     RTSIDE:=RTSIDE-1;
     FOR K:=1 TO SLOPE DO FILL;
   UNTIL (Y>=NEWLEVEL) OR (Y>=FLASKTOP);
   IF COL=BLACK1 THEN DRAWLINE(LTSIDE,Y,RTSIDE,Y,WHITE1); (* display *)
   IF Y>=FLASKTOP THEN
     BEGIN
      NEWLEVEL:=FLASKTOP;
      FILLRATE:=1;    (*flask fills more quickly in the neck *)
     END;

  END

  ELSE  (* Y>=FLASKTOP *)
    BEGIN
     IF Y>=NECKTOP THEN COMMENT
     ELSE
      BEGIN
       WHILE (Y<NECKTOP)AND(Y<=NEWLEVEL) DO FILL;
       IF COL=BLACK1 THEN DRAWLINE(LTSIDE,Y,RTSIDE,Y,WHITE1);
      END;
    END;
 OLDLEVEL:=NEWLEVEL;
END;(* FILLFLASK *)
```

```
(*************************************************************)
PROCEDURE MOVEDROP(*INCR :REAL; VAR LIQLEVEL : INTEGER*);
(*************************************************************)
CONST DELAY=10; SPACING=5;
VAR DROPSIZE : INTEGER;

   (*==========================================================*)
   PROCEDURE DRAWDROP;
   (*==========================================================*)
   VAR  X,STARTDROP,ENDDROP,
        I,MARKTIME: INTEGER;
   BEGIN
    X:=FLASKX;
    STARTDROP:=FLASKY+FLASKSIZ;
    ENDDROP:=STARTDROP-DROPSIZE;
    WHILE ENDDROP>LIQLEVEL DO
     BEGIN
      DRAWLINE(X,STARTDROP,X,ENDDROP,WHITE);(*draw 1 drop*)
      FOR I:=1 TO DELAY DO MARKTIME:=I*I;  (* wait    *)
      DRAWLINE(X,STARTDROP,X,ENDDROP,BLACK); (*erase drop*)
      STARTDROP:=ENDDROP-SPACING;
      ENDDROP:=STARTDROP-DROPSIZE;
     END;
   END; (*DRAWDROP*)

BEGIN  (*MOVEDROP*)
  IF INCR<1.0 THEN DROPSIZE:=1(* select appropriate scaling for vol. of increment*)
       ELSE   DROPSIZE:=ROUND(INCR);
  DRAWDROP;
END; (*MOVEDROP*)

(*************************************************************)
PROCEDURE ACIDMOLARITY(*S:SHORTSTR*);
(*************************************************************)
(* molarity of acid is always displayed at same position - determined by SETUPBOXES *)
BEGIN
 WSTAT(22,178,S)
END;
```

```
(*************************************************************)
PROCEDURE  BASEMOLARITY(*S:SHORTSTR*);
(*************************************************************)
(* molarity of base is always displayed at same position -determined by SETUPBOXES *)
BEGIN
  WSTAT(94.178.S);
END;

(*************************************************************)
PROCEDURE  ACIDISP(*S:SHORTSTR*);
(*************************************************************)
(* volume of acid is always displayed at same position - determined by SETUPBOXES *)
BEGIN
  WSTAT(162.178.S);
END;

(*************************************************************)
PROCEDURE BASEDISP(*S:SHORTSTR*);
(*************************************************************)
(* volume of base is always displayed at the same position - determined by SETUPBOXES*)
BEGIN
  WSTAT(232.178.S);
END;

(*************************************************************)
PROCEDURE  DISPLAYPH(*S:SHORTSTR*);
(*************************************************************)
(* pH is always displayed at the same position - determined by SETUPBOXES*)
BEGIN
  WSTAT(90.145.S);
END;

(*************************************************************)
PROCEDURE  TWOPROMPTS(*S1.S2: STRING*);
(*************************************************************)
(* 2 lines will be displayed at bottom of graphics screen*)
BEGIN
  WSTAT(3.10.S1);
  WSTAT(3,0,S2);
END;


(*$I :TITRAT2*)
(*$I :TITRAT3*)
```

```
(* TITR2.TEXT Included in TITRATLIB. *)
(*************************************************************)
PROCEDURE SETUPCONDITIONS(*VAR HCONC,OHCONC: REAL;
                                        VAR INFLASK: ACIDORBASE*);
(*************************************************************)
VAR CONCSTR : SHORTSTR;

   (*==========================================================*)
   PROCEDURE SELECTCONC(VAR CONC : REAL; VAR ACIDBASE :SHORTSTR);
   (*==========================================================*)
   CONST MIN=0.001; MAX=1.0;     (* Range of soln concentration *)
         X=220; Y=9;             (* coord. to enter input*)
   VAR PROMPT :STRING;  (* string for molarity of acid or base*)
   BEGIN       (* SELECTCONC *)
      PROMPT:='Enter concentration of ';
      PROMPT:=CONCAT(PROMPT,ACIDBASE,':');
      WSTAT(3,Y,PROMPT);           (* DISPLAY PROMPT  *)
      WSTAT(30,0,'(.001M - 1.000M)');
      INRANGERESPONSE(CONC,ACIDBASE,MIN,MAX,X,Y);
      CHARTYPE(6);
      WSTAT(3,Y,PROMPT);            (* ERASE  PROMPT*)
      WSTAT(30,0,'(.001M - 1.000M)');
      CHARTYPE(10);
   END;        (* SELECTCONC *)

   (*==========================================================*)
   PROCEDURE SELECT(VAR INFLASK :ACIDORBASE);
   (*==========================================================*)
   CONST   X=200; Y=0;             (* coord. to enter input *)
   VAR  PROMPT1,PROMPT2:STRING;  REPLY : CHAR;
   BEGIN
      PROMPT1 :='Select solution in flask:';
      PROMPT2:='acid or base     (A/B) ?';
      TWOPROMPTS(PROMPT1,PROMPT2);
      GETHICHAR(X,Y,REPLY,['A','B','Q']);
      WSTAT(X,Y,' '); (* erase char *)
      QUIT:=(REPLY='Q');
      IF REPLY='A' THEN INFLASK:=ACID ELSE INFLASK:=BASE;
      CHARTYPE(6);
      TWOPROMPTS(PROMPT1,PROMPT2); (* ERASE PROMPTS *)
      CHARTYPE(10);
   END; (*SELECT*)

   (*==========================================================*)
   PROCEDURE SELECTVOL(VAR FLASKVOL: REAL);
   (*==========================================================*)
   CONST  MIN=10.0; MAX=50;        (*  range of volume*)
          X=215; Y=10;             (*  coord. to enter input *)
  VAR  PROMPT1,PROMPT2: STRING;
       FLASKSTR: SHORTSTR;
  BEGIN
    IF INFLASK=ACID THEN PROMPT1:='acid' ELSE PROMPT1:='base';
    PROMPT1:=CONCAT('Select vol. of ',prompt1,' in flask:');
    PROMPT2:=' (10.0-50.0mL)';
    TWOPROMPTS(PROMPT1,PROMPT2);
    INRANGERESPONSE(FLASKVOL,FLASKSTR,MIN,MAX,X,Y);
```

```
      CHARTYPE(6);
       TWOPROMPTS(PROMPT1,PROMPT2);  (* ERASE *)
      CHARTYPE(10);
   END; (*SELECTVOL*)

BEGIN        (* SETUPCONDITIONS *)
   CONCSTR:='acid';
   SELECTCONC(HCONC,CONCSTR);
  IF NOT QUIT THEN
   BEGIN
       REALSTR(HCONC,CONCSTR,3,5);
      ACIDMOLARITY(CONCSTR);  (* DISPLAY ACID CONC IN TOP BOX*)
       CONCSTR:='base';
      SELECTCONC(OHCONC,CONCSTR);
       REALSTR(OHCONC,CONCSTR,3,5);
      IF NOT QUIT THEN BASEMOLARITY(CONCSTR);(*DISPLAY BASE CONC IN TOP BOX*)
   END;
  IF TITRTYPE=DIPROTIC THEN INFLASK:=ACID
    ELSE IF NOT QUIT THEN SELECT(INFLASK);
  IF NOT QUIT THEN SELECTVOL(FLASKVOL);
END;        (* SETUPCONDITIONS *)

(***********************************************************)
PROCEDURE CHECKKEY(*VAR SPACEPR,SELECTCHANGE:BOOLEAN*);
(***********************************************************)
VAR SP:CHAR;
BEGIN
    GETACHAR(SP,[SPACE,'C','Q']);
  IF SP=SPACE THEN SPACEPR:=TRUE  ELSE
     BEGIN
        SPACEPR:=FALSE;
        IF SP='C' THEN SELECTCHANGE:=TRUE  ELSE  QUIT:=TRUE;
     END;
END; (*CHECKKEY*)

(************************************************************)
PROCEDURE REQUEST;
(************************************************************)
BEGIN
  WSTAT(2,10,'<SPACE> add increment');
 WSTAT(2,0,'<C>change increment         <Q>quit');
END;  (*REQUEST*)

(***********************************************************)
PROCEDURE INCRPROMPT;
(***********************************************************)
BEGIN
  WSTAT(3,10,'Select titrant increment:');
 WSTAT(3,0,'    (0.05-10.00mL)');
END; (*INCRPROMPT*)
```

```
(****************************************************************)
PROCEDURE SELECTINCR(*VAR INCR :REAL*);
(****************************************************************)
CONST MIN=0.05; MAX=10.00;      (* Range of increments *)
      X=185;Y=9;             (* coord. to enter input *)
VAR  INCSTR: STRING;
BEGIN
  INCRPROMPT;
   INRANGERESPONSE(INCR,INCSTR,MIN,MAX,X,Y);
  CHARTYPE(6);
  INCRPROMPT;   (* ERASE *)
  CHARTYPE(10);
END;(* SELECTINCR *)

(****************************************************************)
PROCEDURE CHANGEINC(*VAR CHANG:BOOLEAN;  VAR INCR:REAL*);
(****************************************************************)
BEGIN
  CHANG:=FALSE;
  CHARTYPE(6);
  REQUEST;        (*erase prompt line*)
  CHARTYPE(10);
  SELECTINCR(INCR);
  REQUEST;        (*display prompt *)
END; (* CHANGEINC*)

(****************************************************************)
PROCEDURE CLEARVALUES (VAR NEWCONC:BOOLEAN);
(****************************************************************)
(*erase all values and flask *)
CONST    BLANK='     ';
BEGIN
   DISPLAYPH(BLANK);          (* erase pH *)
   BASEDISP(BLANK);           (* erase vol. base *)
   ACIDISP(BLANK);            (* erase vol. acid *)
   IF NEWCONC THEN
     BEGIN
        ACIDMOLARITY(BLANK);        (* erase molarity of acid *)
         BASEMOLARITY(BLANK);       (* erase molarity of base *)
      END;
    FILLBOX(10,110,25,125,BLACK1);         (* erase flask  *)
END; (*CLEARVALUES*)

(****************************************************************)
PROCEDURE SETCOLOUR;
(****************************************************************)
 VAR MONITOR:STRING;
 BEGIN
    GETCVAL(MONITOR);
     COLOUR:=(MONITOR='INCOL');
 END;
```

```
(*************************************************************)
PROCEDURE BACKTOMENU;
(*************************************************************)
BEGIN
   WRITE(AT(10,8),'R E L O A D I N G ');
   WRITE(AT(10,11),'M A I N   M E N U . . . . . . ');
END;

(*************************************************************)
PROCEDURE GETK(VAR K1,K2 :REAL);
(*************************************************************)
(* allow user to input pK value(s) *)
VAR   MIN,MAX : REAL;
      OK : BOOLEAN;

    (*=========================================================*)
    PROCEDURE INPUTK(S:STRING; VAR K:REAL);
    (*=========================================================*)
    VAR  PKISTR: SHORTSTR;
         PK:REAL;
         X,Y:INTEGER;

        (*-----------------------------------------------------------*)
        PROCEDURE GETINRANGE(VAR VALUE:REAL; VAR ASTR:SHORTSTR;
                    MIN,MAX:REAL; X,Y:INTEGER);
        (*-----------------------------------------------------------*)
        CONST   MAXLEN=4;
        VAR     RANGE: BOOLEAN;
        BEGIN
          REPEAT
             GETRESPONSE(X,Y,ASTR,MAXLEN,NUMS+['Q']);
            QUIT:=ASTR='Q';
           IF NOT QUIT THEN
            BEGIN
              VALUE:=RVALUE(ASTR);
               RANGE:=(MIN<=VALUE) AND (VALUE<=MAX);
               IF POS('Q',ASTR)>1 THEN RANGE:=FALSE;
               IF (NOT RANGE) THEN  WRITE(AT(X,Y),AROW(MAXLEN,SPACE));
            END;
          UNTIL RANGE OR QUIT;
        END; (*GETINRANGE*)

      BEGIN (* INPUTK*)
        PAGE(OUTPUT);
        X:=0; Y:=4;
         WRITE(AT(X,Y),AROW(40,'*'));  Y:=Y+3;
         WRITE(AT(X+6,Y),'ENTER ',S);  Y:=Y+3;
         WRITE(AT(X,Y),AROW(40,'*'));
        IF TITRTYPE=DIPROTIC THEN WRITE(AT(8,20),'(pK1 MUST BE < pK2)');
         GETINRANGE(PK,PKISTR,MIN,MAX,24,Y-3);
        IF NOT QUIT THEN K:=EXP(-PK*LN(10)) ELSE K:=1.0;
      END; (* INPUTK *)
```

```
BEGIN (*GETK*)
   MIN:=1.0; MAX:=11.0;
   CASE TITRTYPE OF
      WEAKACID : INPUTK('pKa (1-11)',K1);
      WEAKBASE : INPUTK('pKb (1-11)',K1);
      DIPROTIC : BEGIN
                   REPEAT
                      OK:=FALSE;
                      MAX:=11.0;
                      INPUTK('pK1(1-11)',K1);
                      MAX:=13;
                      IF NOT QUIT THEN INPUTK('pK2 (<13)',K2);
                      IF (NOT QUIT) AND (K1<=K2) THEN BEEP ELSE OK:=TRUE;
                    UNTIL (QUIT) OR (OK);
                 END;
    END;(*CASE*)
END;(* GETK *)

(**************************************************************)
PROCEDURE SELECTTYPE(VAR TITRTYPE: TITRAT);
(**************************************************************)
(* User selects type of titration *)
  CONST  STAR='*';
         DOTS='  ....(';        STRONG=' / Strong';
         BASEST=' base';        ACIDST=' acid';
VAR  OK,KOPTION,SOLN:BOOLEAN;
      CH:CHAR;
      X, Y:INTEGER;
      ATYPE,ACH: STRING[10];

    (*============================================================*)
    PROCEDURE WHICHACID(ASOLN:BOOLEAN);
    (*============================================================*)
    VAR
      PROMPT:ARRAY[1..4] OF STRING[20];
      CHOICE:BOOLEAN;

        (*--------------------------------------------------------*)
        PROCEDURE GETREPLY(VAR REPLY :CHAR; ANUM:INTEGER;INPUT:CHARSET);
        (*--------------------------------------------------------*)
         CONST DOTS='......(';
         VAR   J: INTEGER;
               COUNT:STRING[1];
         BEGIN
            COUNT:='';
            PAGE(OUTPUT);
            X:=0; Y:=0;
            WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
            WRITE(AT(X,Y),'Select',ATYPE,' to be used in titration'); Y:=Y+2;
            WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+3;
            FOR J:=1 TO ANUM DO
              BEGIN
                 STR(J,COUNT);
                 WRITE(AT(X,Y),PROMPT[J]);
                 WRITE(AT(X+30,Y),DOTS,COUNT,')'); Y:=Y+3;
              END;
```

```
IF CHOICE THEN
  BEGIN
      STR(ANUM+1,COUNT);
        WRITE(AT(X,Y),'Input pK',ACH,' of',ATYPE);
        WRITE(AT(X+30,Y),DOTS,COUNT,')'); Y:=Y+3;
  END;
 Y:=Y+1;
  WRITE(AT(X+9,Y),'SELECT OPTION');
  WRITE(AT(X+29,Y),DOTS,'    )');
  GETTEXTCHAR(X+37,Y,REPLY,INPUT);
 PAGE(OUTPUT);
END; (*GETREPLY*)

BEGIN (*WHICHACID*)
  CHOICE:=TRUE;
 CASE ASOLN OF
 TRUE : CASE TITRTYPE OF
          STRONGACID,
          WEAKBASE: BEGIN
                         CHOICE:=FALSE;
                          PROMPT[1]:='Hydrochloric';
                          PROMPT[2]:='Nitric';
                          PROMPT[3]:='Perchloric';
                          ATYPE:=ACIDST; ACH:='a ';
                            GETREPLY(CH,3,['1'..'3','Q']);
                       END;
         WEAKACID: BEGIN
                          PROMPT[1]:='Acetic';
                          PROMPT[2]:='Hydrocyanic';
                          PROMPT[3]:='Hydrofluoric';
                          PROMPT[4]:='Benzoic';
                           ATYPE:=' weak acid'; ACH:='a ';
                            GETREPLY(CH,4,['1'..'5','Q']);
                        CASE CH OF
                           '1':  K1:=1.76E-5;
                           '2':  K1:=5.00E-10;
                           '3':  K1:=3.53E-4;
                           '4':  K1:=6.5E-5;
                           '5':  KOPTION:=TRUE;
                         END; (*CASE*)
                       END;
         DIPROTIC: BEGIN
                        PROMPT[1]:='Sulfuric';
                        PROMPT[2]:='Carbonic';
                        PROMPT[3]:='Tartaric';
                        PROMPT[4]:='Oxalic';
                        ATYPE:=ACIDST; ACH:='1 & pK2';
                          GETREPLY(CH,4,['1'..'5','Q']);
                      CASE CH OF
                           '1': BEGIN  K1:=1000;  K2:=1.2E-2;     END;
                           '2': BEGIN  K1:=4.3E-7;  K2:=5.6E-11; END;
                           '3': BEGIN  K1:=1.0E-3; K2:=4.6E-5;    END;
                           '4': BEGIN  K1:=5.9E-2; K2:=6.4E-5;    END;
                           '5': KOPTION:=TRUE;
                         END;(*CASE*)
                     END;
```

```
              END;(*CASE*)
     FALSE: IF  TITRTYPE=WEAKBASE THEN
              BEGIN
                 PROMPT[1]:='Ammonia';
                 PROMPT[2]:='Pyridine';
                 ATYPE:=' weak base'; ACH:='b ';
                  GETREPLY(CH,2,['1'..'3','Q']);
                CASE CH OF
                  '1':  K1:=1.79E-5;
                  '2':  K1:=1.7E-9;
                  '3':  KOPTION:=TRUE;
                 END; (*CASE*)
               END  ELSE
                BEGIN
                   CHOICE:=FALSE;
                    PROMPT[1]:='Sodium hydroxide';
                    PROMPT[2]:='Potassium hydroxide';
                   ATYPE:=BASEST;
                     GETREPLY(CH,2,['1'..'2','Q']);
                END;
      END; (*CASE ASOLN*)
       OK:=CH<>'Q';
    END; (*WHICHACID*)

BEGIN (* SELECTTYPE*)
 REPEAT
     PAGE(OUTPUT);
      X:=0; Y:=0;
       WRITE(AT(X,Y),AROW(40,STAR));Y:=Y+2;
      WRITE(AT(X+10,Y),'TITRATION MENU');    Y:=Y+2;
      WRITE(AT(X,Y),AROW(40,STAR));     Y:=Y+2;
       WRITE(AT(X,Y),'Strong',ACIDST,'     ',STRONG,BASEST,DOTS,'1');     Y:=Y+2;
       WRITE(AT(X,Y),'Weak',ACIDST,'       ',STRONG,BASEST,DOTS,'2');     Y:=Y+2;
       WRITE(AT(X,Y),'Weak',BASEST,'       ',STRONG,ACIDST,DOTS,'3');     Y:=Y+2;
       WRITE(AT(X,Y),'Diprotic',ACIDST,STRONG,BASEST,DOTS,'4');          Y:=Y+2;
     WRITE(AT(X,Y),'Quit - back to MAIN MENU      ',DOTS,'Q');            Y:=Y+3;
      WRITE(AT(X+10,Y),'SELECT TITRATION   ',DOTS,'    ');
        GETTEXTCHAR(X+37,Y,CH,['1'..'4','Q']);
      QUIT:=CH='Q';
     CASE CH OF
         '1': TITRTYPE:=STRONGACID;
         '2': TITRTYPE:=WEAKACID;
         '3': TITRTYPE:=WEAKBASE;
         '4': TITRTYPE:=DIPROTIC;
      END; (* CASE *)
     PAGE(OUTPUT);
      QUIT:=CH='Q';
     IF NOT QUIT THEN
      BEGIN
        SOLN:=TITRTYPE<>WEAKBASE;
        KOPTION:=FALSE;
        WHICHACID(SOLN);
       IF KOPTION THEN
        BEGIN
            GETK(K1,K2);
            IF QUIT THEN OK:=NOT QUIT;
```

```
            KOPTION:=FALSE;
          END;
        IF ((NOT QUIT) AND OK) THEN WHICHACID(NOT SOLN);
        QUIT:=FALSE;
      END;
  UNTIL OK OR QUIT;
END; (* SELECTTYPE *)
```

```
(* TITRAT3.TEXT  Included in TITRATLIB. *)

(**************************************************************)
PROCEDURE CALCPH(*ACIDVOL,BASEVOL,HCONC,OHCONC: REAL;  VAR PH: REAL*);
(**************************************************************)
(* calculates the pH of any soln given the vol of acid & base in  flask and both molarities *)
CONST  DIFF=1.0E-6;                  (*minimum millimoles considered*)
VAR  ACIDMOLS,BASEMOLS,              (*net moles of acid & base in soln*)
     TOTALVOL,                       (*total volume of soln *)
     EXCESS  : REAL;                 (*conc. of excess acid or base*)
     EXCESSACID,EXCESSBASE,EQUIVPT :BOOLEAN;   (*nature of soln*)

    (*==========================================================*)
    PROCEDURE NEWTON(A,B,C,D,E,APPROX:REAL;  VAR PH:REAL);
    (*==========================================================*)
    CONST CRITERIA :=0.001 ;
    VAR  COUNT : INTEGER;
          NEWTONX,ERROR, GUESS  : REAL;
          SOLN : BOOLEAN;

       (*-------------------------------------------------------*)
       FUNCTION EQUATION(X:REAL):REAL;
       (*-------------------------------------------------------*)
       BEGIN
          EQUATION:=E+X*(D + X*(C+X*(B+A*X)));
       END;

       (*-------------------------------------------------------*)
       FUNCTION DERIV(X:REAL):REAL;
       (*-------------------------------------------------------*)
       BEGIN
          DERIVE:= D+X*(2*C + X(3*B + 4*A*X));
       END;

    BEGIN  (*NEWTON*)
    COUNT :=0;
    GUESS :=APPROX;
    SOLN :=FALSE;
    REPEAT
      COUNT :=COUNT+1 ;
      NEWTONX:=APPROX-(EQUATION(APPROX)/DERIV(APPROX));
      IF ABS((NEWTONX-APPROX)/NEWTONX) < CRITERIA THEN SOLN:=TRUE
         ELSE  APPROX:=NEWTONX;
     UNTIL ((COUNT>20) OR (SOLN));
     IF (NEWTONX<0) OR (NOT SOLN) THEN
      BEGIN
         APPROX:=GUESS*10;
         NEWTON(A,B,C,D,E,APPROX);
      END    ELSE  PH:=-LOG(NEWTONX);
    END;  (*NEWTON*)
```

```
(*=================================================================*)
PROCEDURE SOLVEQN(ACID,SALT1,SALT2,GUESS:REAL;VAR PH:REAL);
(*=================================================================*)
CONST  POWER= 1E5;
VAR   A : ARRAY [1..5] OF REAL;
      NUM,SCALE : INTEGER;
BEGIN
 A[1]:=1.0
 A[2]:= K1 +  SALT1  +  2*SALT2;
 A[3]:= K1*K2 - K1*ACID + K1*SALT2 - KW;
 A[4]:= -(K1*KW    + 2*K1*K2*ACID  + K1*K2*SALT1)
 A[5]:= -(K1*K2*KW);
 FOR NUM:=2 TO 5 DO
    FOR SCALE:=1 TO NUM-1 DO   A[NUM]:=A[NUM]*POWER;
 GUESS:=GUESS*POWER;
 IF TITRTYPE=DIPROTIC THEN
     NEWTON(A[1],A[2],A[3],A[4],A[5],GUESS,H)
    ELSE  NEWTON(0,A[1],A[2],A[3],A[4],GUESS,H);
 H:=(H/POWER);
END; (*SOLVEQN*)

(*=================================================================*)
PROCEDURE STRONGCALC;
(*=================================================================*)
CONST  DILUTE=1.0E-6;

   (*--------------------------------------------------------------*)
   PROCEDURE CORRECT(VAR VALUE:REAL);
   (*--------------------------------------------------------------*)
   (* Consider contribution of ions from the dissociation  of water*)
   VAR DISCRIM: REAL;
   BEGIN
     DISCRIM:=SQRT((VALUE*VALUE) + 4*KW);
     VALUE:=(VALUE+DISCRIM) / 2;
   END; (*CORRECT*)

BEGIN
  IF EXCESS<DILUTE THEN CORRECT(EXCESS)
  PH:=-LOG(EXCESS)
  IF EXCESSBASE THEN PH:=14-PH;
END;(* STRONGCALC*)

(*=================================================================*)
PROCEDURE WEAKCALC;
(*=================================================================*)
(*Calculates pH of soln of weak acid/strong base of weak base/ strong acid.  Acid may
be in either flask or burette*)
VAR  SALT1CONC,SALT2CONC,APPROXH:REAL;
      WEAKEXCESS:BOOLEAN;

   (*--------------------------------------------------------------*)
   PROCEDURE HYDROLYSIS(SALT:REAL; VAR H,PH:REAL);
   (*--------------------------------------------------------------*)
   CONST  DILUTE=1.0E-6;
   VAR SALTHYD:REAL;
   BEGIN
```

```
              ACIDCONC:=EXCESS;
              SALT1CONC:=BASEMOLS/TOTALVOL;
              SALT2CONC:=0.0;
              IF SALT1CONC=0 THEN APPROXH:=SQRT(K1*ACIDCONC)
                ELSE IF ACIDCONC:=0 THEN APPROXH:=AQRT(K1*K2)
                 ELSE  APPROXH:=K1*ACIDCONC/SALT1CONC;
           END;
         SOLVEQN(ACIDCONC,SALT1CONC,SALT2CONC,APPROXH,PH);
       END;
    END;  (*DICALC*)


BEGIN  (*CALCPH*)
   TOTALVOL:=ACIDVOL+BASEVOL;
   ACIDMOLS:=ACIDVOL*HCONC;
   BASEMOLS:=BASEVOL*OHCONC;
   EXCESS:=ACIDMOLS-BASEMOLS;
   IF ABS(EXCESS)<DIFF THEN EXCESS:=0.0;
   EXCESSACID:=EXCESS>0;
   EXCESSBASE:=EXCESS<0;
   EQUIVPT:=EXCESS=0;
   EXCESS:=ABS(EXCESS)/TOTALVOL;
   CASE TITRTYPE OF
      STRONGACID: STRONGCALC;
      WEAKACID,
      WEAKBASE:  WEAKCALC;
      DIPROTIC:  DICALC;
   END; (* CASE *)
END;  (* CALCPH *)


BEGIN
    NUMS:=['.','0'..'9'];
END.
```

```
(*=================================================================*)
PROCEDURE BORDER(COL :SCREENCOLOR);
(*=================================================================*)
CONST WIDTH=10;
BEGIN
 FILLBOX(XMIN,XMAX,YMIN,YMIN+WIDTH,COL);
 FILLBOX(XMIN,XMAX,YMAX-WIDTH,YMAX,COL);
 FILLBOX(XMIN,XMIN+WIDTH,YMIN,YMAX,COL);
 FILLBOX(XMAX-WIDTH,XMAX,YMIN,YMAX,COL);
END;

(*=================================================================*)
 PROCEDURE INITCONDITIONS;
(*=================================================================*)
 VAR ASTR: SHORTSTR;
 BEGIN
  FILLRATE :=2;    (*determines rate at which flask filled  *)
 END;  (*INITCONDITIONS*)

(*=================================================================*)
 PROCEDURE INITLEVEL;
(*=================================================================*)
 (*initializes coord of sides of flask & top of flask as well as level of soln in flask *)
 BEGIN
  FLASKTOP :=FLASKY+(3*FLASKSIZ)DIV 4; (*ycoord of top sloping sides of flask*)
  NECKTOP :=FLASKY+FLASKSIZE;      (*ycoord of very top of flask*)
  LTSIDE :=FLASKX-(FLASKSIZ DIV 2)+2; (*calc. coord of sides of*)
  RTSIDE :=FLASKX+(FLASKSIZ DIV 2)-2; (*flask given midpt of base *)
  OLDLEVEL :=FLASKY+1;             (*base of flask - flasky    *)
  INCREASE :=26;  (*depth of soln to be initially placed in flask*)
  XTRAVOL :=0.0; (*initialize increment in titrant to be displayed*)
 END;(*INITLEVEL*)

(*=================================================================*)
 PROCEDURE CHANGECOL(NEWCOLOR :SCREENCOLOR);
(*=================================================================*)
 (*change colour of soln to newcolour  *)
 VAR CURRENTL : INTEGER;
 BEGIN
  SOLNCOL :=NEWCOLOR;
  CURRENTL :=OLDLEVEL;
  INITLEVEL;
  CURRENTL :=CURRENTL-OLDLEVEL;
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,CURRENTL,NEWCOLOR);
 END; (*CHANGECOL*)
```

```
BEGIN (* TITLEPAGE *)
  INITCONDITIONS;
  INITLEVEL;
 DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE2);
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
 X:=150;
 BORDER(VIOLET);
 WSTAT(130,150,'A C I D / B A S E');
 WSTAT(123,130,'T I T R A T I O N S');
 WSTAT(X,71,'By');
 WSTAT(X,49,'Roslyn Atkins,');
 WSTAT(X,37,'Chemistry Dept.');
 WSTAT(X,25,'Wollongong Uni.');
 WRITE('PROC. TITLEPAGE ',MEMAVAIL);

 WAIT(30000);
 GRAFMODE;
 L:=0;
 REPEAT
   L:=L+1;
   CASE L OF
    1,6,11 : NEWCOL:=BLUE;
    2,7,12 : NEWCOL:=WHITE2;
    3,8,13 : NEWCOL:=ORANGE;
    4,9,14 : NEWCOL:=WHITE2;
    5,10,15 :NEWCOL:=VIOLET;
   END; (*CASE*)
   CHANGECOL(NEWCOL);
   WAIT(250);
 UNTIL (L=15) OR (KEYIN);
  IF KEYIN THEN READ(CH);
END; (* TITLEPAGE*)

(**********************************************************)
PROCEDURE GETCOLOUR;
(**********************************************************)
CONST  XX=60; YY=80; WIDTH=20;
VAR   X,Y: INTEGER; CH:CHAR;   MONITOR:STRING;
BEGIN
   X:=XMIN+XX; Y:=YMAX-YY;
   WSTAT(X,Y,'Are you using a ');
   WSTAT(X,Y-20,'colour monitor?(Y/N)');
   GETHICHAR(220,Y-20,CH,['Y','N']);
   IF CH='Y' THEN MONITOR:='INCOL' ELSE MONITOR:='NOCOL';
   SETCVAL(MONITOR);
   FILLBOX(XMIN+WIDTH,XMAX-WIDTH,YMIN+WIDTH,YMAX-WIDTH,BLACK1);
END; (* GETCOLOUR *)
```

```
(*****************************************************************)
PROCEDURE PRESSRETURN;
(*****************************************************************)
CONST XX=30; YY=50; WIDTH=20;
VAR   X,Y: INTEGER; CH:CHAR;
BEGIN
   X:=XMIN+XX; Y:=YMAX-YY;
   WSTAT(X,Y,'Throughout these programs');
   WSTAT(X,Y-18,'any data that you enter');
   WSTAT(X,Y-36,'must be followed by ');
   WSTAT(X,Y-54,'the <RETURN> key.');
   WSTAT(X,Y-105,'Press <SPACE BAR> to continue');
   GETACHAR(CH,[SPACE]);
   FILLBOX(XMIN+WIDTH,XMAX-WIDTH,YMIN+WIDTH,YMAX-WIDTH,BLACK1);
END; (* PRESSRETURN *)

(*****************************************************************)
PROCEDURE HOWTOQUIT;
(*****************************************************************)
CONST XX=50; YY=80;
VAR   X,Y: INTEGER; CH:CHAR;
BEGIN
   X:=XMIN+XX; Y:=YMAX-YY;
   WSTAT(X,Y,'To exit this program'); Y:=Y-20;
   WSTAT(X,Y,'at any time input "Q"'); Y:=Y-75;
   WSTAT(X,Y,'Press <SPACE BAR> to continue');
   GETACHAR(CH,[SPACE,'Q']);
   QUIT:=CH='Q';
   PAGE(OUTPUT);
END; (* HOWTOQUIT *)

(*****************************************************************)
PROCEDURE FIN;
(*****************************************************************)
 VAR  X,Y:INTEGER;
 BEGIN
    PAGE(OUTPUT);
    X:=5; Y:=6;
    WRITE(AT(X+2,Y),'REMOVE DISK FROM DISK DRIVE');  Y:=Y+5;
    WRITE(AT(X,Y),'IT WILL BE NECESSARY TO REBOOT'); Y:=Y+2;
    WRITE(AT(X,Y),'COMPUTER TO RUN ANOTHER PROGRAM   ');
    REPEAT
       X:=Y;   (* INFINITE LOOP *)
    UNTIL (X>Y);
 END; (* FIN *)

(*****************************************************************)
PROCEDURE CHAINTOMENU;
(*****************************************************************)
BEGIN
   SETCHAIN(':MENU');
   GOTOXY(0,10);
   WRITE('L O A D I N G   M E N U ....... ');
END; (*CHAINTOMENU*)
```

```
BEGIN (* MAIN *)
   INITTURTLE;
   TEXTMODE;
   TITLEPAGE;
   ENCLOSE(VIOLET);
   PRESSRETURN;
   GETCOLOUR;
   HOWTOQUIT;
   TEXTMODE;
   IF QUIT THEN FIN  ELSE CHAINTOMENU;
END. (*INTRO*)
```

```
(* Main menu for ACID/BASE TITRATION PACKAGE which chains to
           - titration of acids & bases  (TITRATE)
           - titration of acids & bases with indicators(INDICATOR)
           - titration of salts  (SALTTITRATE)
           - titration quiz   (QUIZ)
           - titration assignment (ASSIGNMENT)                 *)
(*$S++,R-,V-*)

PROGRAM MENU;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;
CONST
 XCON=160; YCON=40;
 FLASKX=60; FLASKY=24; FLASKSIZ=100;
 STAR='*';
VAR
 ACH:CHAR;
 READY:BOOLEAN;
 PROGNUM:CHAR;
 PICT,COLOUR:BOOLEAN;
 QUIT: BOOLEAN;

(******************************************************************)
PROCEDURE DOUBLELINE(X1,Y1,X2,Y2:INTEGER; COL: SCREENCOLOR);
(******************************************************************)
BEGIN
   DRAWLINE(X1,Y1,X2,Y2,COL);
   DRAWLINE(X1+1,Y1+1,X2+1,Y2+1,COL);
END;

(******************************************************************)
PROCEDURE DRAWBOX(X1,Y1,X2,Y2: INTEGER; COL: SCREENCOLOR);
(******************************************************************)
(* x1,y1 are coord. of bottom left hand corner & x2,y2 coord. of top right hand corner of
box to be displayed *)
VAR  DOUBLE : INTEGER;
BEGIN
 FOR DOUBLE:=1 TO 2 DO
    BEGIN
      MOVECOL(x1,y1,col);
      MOVETO(x2,y1);
      MOVETO(x2,y2);
      MOVETO(x1,y2);
      MOVECOL(x1,y1,NONE);
      X1:=X1+1;X2:=X2+1;Y1:=Y1+1;Y2:=Y2+1;
    END;
END;

(******************************************************************)
PROCEDURE INITSCREEN;
(******************************************************************)

   (*=============================================================*)
   PROCEDURE DRAWBURETTE(X,Y,SIZE: INTEGER; COL : SCREENCOLOR);
   (*=============================================================*)
   VAR  WIDTH,LENGTH : INTEGER;
```

```
(*---------------------------------------------------------------------------*)
PROCEDURE ONESIDE(X1 ,Y1 ,WWIDTH,LENGTH : INTEGER);
(*---------------------------------------------------------------------------*)
BEGIN
   X1 :=X1-WWIDTH;
    MOVECOL(X1 ,Y1 ,COL);
    Y1 :=Y1+LENGTH;
    MOVETO(X1 ,Y1);
    Y1 :=Y1+ABS(2*WWIDTH);
    X1 :=X1-2*WWIDTH;
    MOVETO(X1 ,Y1);
    MOVECOL(X1 ,Y+SIZE,NONE);
END; (*ONESIDE*)

(*---------------------------------------------------------------------------*)
PROCEDURE TAP;
(*---------------------------------------------------------------------------*)
BEGIN
   DRAWBOX(X-WIDTH,Y+LENGTH-6,X+WIDTH,Y+LENGTH-2,COL);
   FILLBOX(X-WIDTH+1 ,X+WIDTH-1 ,Y+LENGTH-5,Y+LENGTH-3,BLACK2);
END;(*TAP*)

BEGIN  (* DRAWBURETTE*)
 WIDTH:=SIZE DIV 8;
 LENGTH:=SIZE DIV 3;
 ONESIDE(X,Y,WIDTH DIV 2,LENGTH);
 ONESIDE(X,Y,-WIDTH DIV 2,LENGTH);
 TAP;
END;  (* DRAWBURETTE*)

(*===========================================================================*)
PROCEDURE SETUPBOXES(COL : SCREENCOLOR);
(*===========================================================================*)
CONST   LOWERY=170;   TOPY=190;    (* y coord. of small boxes *)

  (*-------------------------------------------------------------------------*)
  PROCEDURE LABELS(X: INTEGER; CH1 ,CH2:SHORTSTR);
  (*-------------------------------------------------------------------------*)
  CONST  HEIGHT=179; (*y coord of title of small boxes *)
  BEGIN
     WSTAT(X,HEIGHT,CH1);
     WSTAT(X+7,HEIGHT-7,CH2);
  END; (*LABELS*)

BEGIN (*SETUPBOXES *)
  DRAWBOX(1 ,20,135,165,COL);          (* main left box *)
  DRAWBOX(143,20,275,165,COL);         (* main right box*)
  DRAWBOX(1 ,LOWERY,64,TOPY,COL);      (* acid molarity *)
  LABELS(7,'M','a');
  DRAWBOX(70,LOWERY,135,TOPY,COL);     (* base molarity *)
  LABELS(78,'M','b');
  DRAWBOX(143,LOWERY,206,TOPY,COL);    (* acid volume   *)
  LABELS(149,'V','a');
  DRAWBOX(212,LOWERY,275,TOPY,COL);    (* base volume   *)
  LABELS(218,'V','b');
  FILLBOX(84,135,134,165,COL);         (* set up pH box *)
```

```
    FILLBOX(88,130,138,162,BLACK2);
  END; (* SETUPBOXES*)

BEGIN      (* INITSCREEN *)
  INITTURTLE;
  TEXTMODE;
  SETUPBOXES(BLUE); (*to hold molarity & vol of acid & base& pH*)
  DRAWBURETTE(FLASKX,FLASKY+FLASKSIZE,38,WHITE2);
  (* burette bottom  centre of tip  at x,y  of length  *)
END; (* INITSCREEN*)

(*************************************************************)
PROCEDURE DRAWFLASK(X,Y,SIZE : INTEGER; COL :SCREENCOLOR);
(*************************************************************)
VAR RUN,RISE,WIDTH : INTEGER;

  (*=========================================================*)
  PROCEDURE ONESIDE(X1,Y1 : INTEGER);
  (*=========================================================*)
  BEGIN
    MOVECOL(X1,Y1,COL);
    MOVETO(X1+RUN,Y+RISE);
    MOVECOL(X1+RUN,Y+SIZE,NONE);
  END; (*ONESIDE*)

BEGIN (* DRAWFLASK *)
  WIDTH:=SIZE DIV 2;
  RUN:=3*SIZE DIV 8;
  RISE:=2*RUN;
  ONESIDE(X-WIDTH,Y);
  RUN:=-RUN;
  ONESIDE(X+WIDTH,Y);
  DRAWLINE(X+WIDTH,Y,X-WIDTH,Y,COL);
END; (* DRAWFLASK *)

(*************************************************************)
PROCEDURE DRAWAXES(X,Y,SIZE : INTEGER; COL :SCREENCOLOR);
(*************************************************************)
BEGIN
  WSTAT(X-10,Y-2+(SIZE DIV 2),'7-');
  WSTAT(X-8,Y+SIZE+2,'pH');
  WSTAT(X+SIZE-15,Y-9,'Vol');
  DRAWLINE(X,Y+SIZE,X,Y,COL);        (* y coord of graph*)
  DRAWLINE(X,Y,X+SIZE,Y,COL);        (* x coord of graph*)
END;

(*************************************************************)
PROCEDURE TWOPROMPTS(S1,S2 : STRING);
(*************************************************************)
BEGIN
  WSTAT(3,10,S1);
  WSTAT(3,0,S2);
END;
```

```
(*****************************************************************)
PROCEDURE GETSPACEBAR;
(*****************************************************************)
VAR CH: CHAR;
BEGIN
   WRITE(AT(23,23),'press <SPACE BAR>');
   GETACHAR(CH,[SPACE,'Q']);
   QUIT :=(CH='Q');
END;

(*****************************************************************)
PROCEDURE SELECTOPTION(Y:INTEGER; VAR CH:CHAR; LEGALSET:CHARSET);
(*****************************************************************)
BEGIN
   WRITE(AT(0,Y),'        SELECT OPTION  ..........(  )');
   GETTEXTCHAR(37,Y,CH,LEGALSET);
   QUIT :=CH='Q';
END;

(*****************************************************************)
PROCEDURE SHOWMENU(VAR NUM: CHAR);
(*****************************************************************)
  CONST  DOTS='     .....(';      TITR='Titration ';
    AORB='of acids & bases';   ST=' students';  BLANK='      '; X=0;
VAR  Y : INTEGER;
BEGIN
   PAGE(OUTPUT); Y:=0;
   WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
   WRITE(AT(X+10,Y),'M A I N   M E N U'); Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+3;
   WRITE(AT(X,Y),TITR,'of acids & bases',DOTS,'1)'); Y:=Y+2;
   WRITE(AT(X,Y),TITR,'of acids & bases'); Y:=Y+1;
   WRITE(AT(X,Y),'  using indicators ',BLANK,DOTS,'2) '); Y:=Y+2;
   WRITE(AT(X,Y),TITR,'quiz        ',BLANK,DOTS,'3)'); Y:=Y+2;
   WRITE(AT(X,Y),TITR,'assignment',BLANK,DOTS,'4)'); Y:=Y+2;
   WRITE(AT(X,Y),'QUIT               ',BLANK,DOTS,'Q)'); Y:=Y+3;
   SELECTOPTION(Y,NUM,['1'..'4','Q']);
   PICT :=((NUM='1') OR (NUM='2'));
END;  (* SHOWMENU *)

(*****************************************************************)
 PROCEDURE STARTPROG(VAR REPLY:CHAR);
(*****************************************************************)
 CONST X=0;  BLANK='            ....(';
 VAR Y:INTEGER;
 BEGIN
    PAGE(OUTPUT);
    Y:=7;
    WRITE(AT(X,Y),'Repeat instructions',BLANK,'R)'); Y:=Y+2;
    WRITE(AT(X,Y),'Start program      ',BLANK,'S)'); Y:=Y+2;
    WRITE(AT(X,Y),'Back to MAIN MENU  ',BLANK,'M)'); Y:=Y+4;
    SELECTOPTION(Y,REPLY,['R','S','M','Q']);
    PAGE(OUTPUT);
    READY:=REPLY='S';
 END; (* STARTPROG*)
(*$I :MENUB*)
```

```
(* MENUB - INCLUDED IN MENU *)
(***************************************************************)
PROCEDURE TEXTINTRO;
(***************************************************************)
VAR   CH:CHAR; X,Y:INTEGER;
      S:ARRAY[1..5] OF STRING[40];

   (*==========================================================*)
   PROCEDURE HEADING;
   (*==========================================================*)
   BEGIN
     PAGE(OUTPUT);
     X:=0; Y:=0;
     WRITE(AT(X,Y),AROW(40,STAR));  Y:=Y+2;
     WRITE(AT(8,Y),'I N S T R U C T I O N S'); Y:=Y+2;
     WRITE(AT(X,Y),AROW(40,STAR));  Y:=Y+3;
   END;

   (*==========================================================*)
   PROCEDURE WRITESTR(STRNUM:INTEGER);
   (*==========================================================*)
   VAR J: INTEGER;
   BEGIN
     S[1]:=CONCAT('** ',S[1]);
     WRITE(AT(0,Y),S[1]);  Y:=Y+1;
     FOR J:=2 TO STRNUM DO
       BEGIN
        WRITE(AT(3,Y),S[J]);   Y:=Y+1;
       END;
     Y:=Y+2;
   END;

   (*==========================================================*)
   PROCEDURE WORKSHEETS;
   (*==========================================================*)
   BEGIN
     S[1]:='WORKSHEETS are available to guide';
     S[2]:='you through these demonstrations.';
     WRITESTR(2);
   END;

   (*==========================================================*)
   PROCEDURE SUGGEST(ASTR1,ASTR2:STRING);
   (*==========================================================*)
   BEGIN
    S[1]:='It is suggested that you work';
    S[2]:=CONCAT('through the ',ASTR1,' program');
    S[3]:=CONCAT('before attempting this ',ASTR2,'.') ;
    WRITESTR(3);
   END;
```

```
(*=============================================================*)
PROCEDURE ASSIG1;
(*=============================================================*)
BEGIN
   S[1]:='Use this program at the direction';
   S[2]:='of your TEACHER.';
   S[3]:='(Answers are in the Teachers Manual)';
   WRITESTR(3);
END;

(*=============================================================*)
PROCEDURE ASSIG2;
(*=============================================================*)
BEGIN
   S[1]:='The answers to these assignments';
   S[2]:='will NOT be supplied by the computer.';
   S[3]:='They must be handed in to your';
   S[4]:='TEACHER for assessment.';
   WRITESTR(4);
END;

(*=============================================================*)
PROCEDURE ASSIG3;
(*=============================================================*)
BEGIN
   S[1]:='It will be necessary to enter';
   S[2]:='an ASSIGNMENT NUMBER (1-99).';
   S[3]:='This number will be given ';
   S[4]:='to you by your teacher.';
   WRITESTR(4);
END;

(*=============================================================*)
PROCEDURE QUIZ1;
(*=============================================================*)
BEGIN
   S[1]:='The computer will select';
   S[2]:='the concentration of an acid or';
   S[3]:='base - YOU have to DETERMINE this';
   S[4]:='concentration by carrying out a';
   S[5]:='simulated titration.';
   WRITESTR(5);
END;

(*=============================================================*)
PROCEDURE TERMINATE;
(*=============================================================*)
BEGIN
  S[1]:='To TERMINATE the titration ';
  S[2]:='at any time enter "Q" ';
  WRITESTR(2);
END;
```

```
(*=============================================================*)
PROCEDURE HYPOIND;
(*=============================================================*)
BEGIN
  S[1]:='These titrations use a HYPOTHETICAL';
  S[2]:='INDICATOR which will change colour';
  S[3]:='exactly at the equivalence point.';
  WRITESTR(3);
END;

(*=============================================================*)
PROCEDURE SLOWER;
(*=============================================================*)
BEGIN
  S[1]:='Titrations involving DILUTE solutions';
  S[2]:='are SLOWER than more concentrated';
  S[3]:='solutions.';
  WRITESTR(3);
END;

(*=============================================================*)
PROCEDURE COLPROB;
(*=============================================================*)
BEGIN
  S[1]:='Due to limitation of colour graphics';
  S[2]:='the colours displayed in this ';
  S[3]:='program are NOT exactly the same';
  S[4]:='as the true INDICATOR COLOURS.';
  S[5]:='(See manual for further details)';
  WRITESTR(5);
END;

(*=============================================================*)
PROCEDURE NOCOL;
(*=============================================================*)
BEGIN
  S[1]:='Since a colour monitor is not';
  S[2]:='available, all indicator changes';
  S[3]:='will appear as BLACK to WHITE';
  S[4]:='or WHITE to BLACK.';
  S[5]:='(See manual for true colours)';
  WRITESTR(5);
END;

(*=============================================================*)
PROCEDURE DESCRIB;
(*=============================================================*)
BEGIN
  S[1]:='This program will allow you to';
  S[2]:='INVESTIGATE the relationship';
  S[3]:='between pH and other titration';
  S[4]:='variables.';
  WRITESTR(4);
END;
```

```
(*==================================================================*)
PROCEDURE INDIC;
(*==================================================================*)
BEGIN
  S[1]:='This program will allow you to';
  S[2]:='DETERMINE the most APPROPRIATE';
  S[3]:='INDICATOR for a given titration.';
  WRITESTR(3);
END;

(*==================================================================*)
PROCEDURE NEWPAGE;
(*==================================================================*)
BEGIN
  WRITE(AT(3,23),'For further details');
  GETSPACEBAR;
  IF QUIT THEN EXIT(TEXTINTRO) ELSE HEADING;
  Y:=Y+1
END;

BEGIN (*TEXTINTRO*)
 HEADING;
 CASE PROGNUM OF
 '1': BEGIN
        WORKSHEETS;
        DESCRIB;
        HYPOIND;
        NEWPAGE;
        SLOWER;
     END;
 '2': BEGIN
        WORKSHEETS;
        INDIC;
        IF COLOUR THEN COLPROB ELSE NOCOL;
        NEWPAGE;
        SLOWER;
     END;
 '3': BEGIN
        SUGGEST('TITRATION','QUIZ');
        QUIZ1;
        NEWPAGE;
        HYPOIND;
     END;
 '4': BEGIN
        SUGGEST('QUIZ','ASSIGNMENT');
        ASSIG1;  ASSIG2; NEWPAGE;
        QUIZ1;
        ASSIG3; NEWPAGE;
        HYPOIND;
     END;
 END; (*CASE*)
 TERMINATE;
 IF PICT THEN WRITE(AT(3,23),'For further details');
 GETSPACEBAR;
 PAGE(OUTPUT);
END;(* TEXTINTRO*)
```

```
(****************************************************************)
 PROCEDURE GRAPHINTRO;
(****************************************************************)

   (*============================================================*)
   PROCEDURE GRAPH;
   (*============================================================*)
   CONST  M='Molarity of '; V='Volume of ';
          A='acid'; B='base'; F=' in flask';
          GETSPACE='Press <SPACE BAR>';
          SHOWN=' will be shown';
          INBOX='in this box.        Press <SPACE BAR>';
   VAR  J: INTEGER;
       S1,S2: STRING;  CH:CHAR;

      (*----------------------------------------------------------*)
      PROCEDURE FLASH(NUM:INTEGER);
      (*----------------------------------------------------------*)
      VAR  X,Y,J,WIDTH : INTEGER;
           CH:CHAR;
      BEGIN
       CASE NUM OF
         1: BEGIN  X:=23;  Y:=178; END;
         2: BEGIN  X:=96;  Y:=178; END;
         3: BEGIN  X:=168; Y:=178; END;
         4: BEGIN  X:=238; Y:=178; END;
         5: BEGIN  X:=98;  Y:=145; END;
         6: BEGIN  X:=190; Y:=90; END;
         7: BEGIN  X:=XCON-17; Y:=25;  END;
         8: BEGIN  X:=190; Y:=60;  END;
         9: BEGIN  X:=190; Y:=110; END;
        10:BEGIN  X:=190; Y:=85;  END;
       END; (* CASE *)
       X:=X+12;
       CHARTYPE(3);
       CH:='X';
       REPEAT
          WSTAT(X,Y,SPACE); (* display *)
          WAIT(80);
          WSTAT(X,Y,SPACE); (* erase   *)
          IF KEYIN THEN READ(KEYBOARD,CH);
          IF ((CH='Q') OR (CH='q')) THEN EXIT(GRAPH);
          WAIT(5);
       UNTIL CH=SPACE;
       CHARTYPE(6);
      END; (* FLASH *)

      (*----------------------------------------------------------*)
      PROCEDURE TWOPR(NUM:INTEGER; S1,S2:STRING);
      (*----------------------------------------------------------*)
      BEGIN
       TWOPROMPTS(S1,S2);
       FLASH(NUM);
       TWOPROMPTS(S1,S2); (*ERASE*)
      END; (* TWOPR *)
```

```
(*-----------------------------------------------------------------------*)
PROCEDURE EXTRA;
(*-----------------------------------------------------------------------*)
     PROCEDURE SHOWRANGE;
     VAR   START,FIN, UPPER,LOWER:INTEGER;
     BEGIN
      START:=XCON+2; FIN:=XCON+100;
      LOWER:=YCON+40; UPPER:=YCON+60;
      FILLBOX(START,FIN,YCON+2,LOWER,VIOLET);
      FILLBOX(START,FIN,UPPER,YCON+100,BLUE);
      DRAWLINE(START,LOWER,FIN,LOWER,WHITE2);
      DRAWLINE(START,UPPER,FIN,UPPER,WHITE2);
     END; (*SHOWRANGE*)

 BEGIN (* EXTRA *)
  S1:=CONCAT('Indicator selected',SHOWN);
  S2:=CONCAT('here.             ',GETSPACE);
  TWOPR(7,S1,S2);
  SHOWRANGE;
  S1:='pH range in which indicator';
  S2:=CONCAT('is a certain colour.   ',GETSPACE);
  TWOPR(8,S1,S2);
  S2:=CONCAT('is a different colour. ',GETSPACE);
  TWOPR(9,S1,S2);
  S2:=CONCAT('changes colour.        ',GETSPACE);
  TWOPR(10,S1,S2);
 END; (*EXTRA*)

BEGIN (*GRAPH*)
  FILLBOX(XMIN,XMAX,YMIN,18,BLACK); (*Clear any old prompts*)
  IF PROGNUM='2' THEN  WSTAT(XCON,YCON-15,'Indicator');
  S1:='The following titration data ';
  S2:='will be displayed on screen';
  TWOPROMPTS(S1,S2);
  FOR J:=1 TO 7 DO WAIT(30000);
  TWOPROMPTS(S1,S2);
  TWOPR(1,CONCAT(M,A,SHOWN),INBOX);
  TWOPR(2,CONCAT(M,B,SHOWN),INBOX);
  TWOPR(3,CONCAT(V,A,F,SHOWN),INBOX);
  TWOPR(4,CONCAT(V,B,F,SHOWN),INBOX);
  TWOPR(5,CONCAT('pH of solution',F,SHOWN),INBOX);
  S1:='pH vs volume of titrant added will be';
  S2:=CONCAT('graphed.        ',GETSPACE);
  TWOPR(6,S1,S2);
  IF PROGNUM='2' THEN EXTRA;
 END; (* GRAPH*)


BEGIN (* GRAPHINTRO*)
  GRAFMODE; (* set monitor to graphics mode*)
  CHARTYPE(6);
  GRAPH;
  CHARTYPE(10);
  TEXTMODE;
END;(*GRAPHINTRO*)
```

```
(**************************************************************)
PROCEDURE INTROTITRATE;
(**************************************************************)
VAR REPLY:CHAR;
BEGIN
 REPEAT
  TEXTINTRO;
  IF NOT QUIT THEN
   BEGIN
    IF PICT THEN GRAPHINTRO;
   END;
  STARTPROG(REPLY);
 UNTIL ((QUIT OR READY) OR (REPLY='M'));
END; (* INTROTITRATE *)

(**************************************************************)
PROCEDURE INSTRUCT;
(**************************************************************)
CONST X=0;
VAR  Y:INTEGER;
     CH:CHAR;
BEGIN
  PAGE(OUTPUT);
  Y:=10;
  WRITE(AT(X,Y),'Do you want instructions for'); Y:=Y+2;
  WRITE(AT(X,Y),'this program? (y/n) ');
  GETTEXTCHAR(X+20,Y,CH,['Y','N','Q']);
  READY:=CH='N';
  QUIT:=CH='Q';
  IF CH='Y' THEN INTROTITRATE;
END; (* INSTRUCT *)

(**************************************************************)
PROCEDURE CHAINTO(CH: CHAR);
(**************************************************************)
 VAR S:STRING;

   (*==========================================================*)
    PROCEDURE INFORM(NAME:STRING);
   (*==========================================================*)
    BEGIN
      PAGE(OUTPUT);
      WRITE(AT(8,8),'L O A D I N G');
      WRITE(AT(0,12),NAME,'  P R O G R A M . . . . .');
    END;

BEGIN (* CHAINTO *)
  CASE CH OF
   '1': BEGIN S:='T I T R A T I O N';   SETCHAIN(':TITRATE'); END;
   '2': BEGIN S:='I N D I C A T O R S'; SETCHAIN(':INDICATOR'); END;
   '3': BEGIN S:='    Q U I Z  ';       SETCHAIN(':QUIZ');     END;
   '4': BEGIN S:='A S S I G N M E N T'; SETCHAIN(':ASSIGNM');  END;
    END; (*CASE*)
  INFORM(S);
END; (* CHAINTO *)
```

```
(****************************************************************)
PROCEDURE CHECKCOL;
(****************************************************************)
VAR  MONITOR:STRING;
BEGIN
   GETCVAL(MONITOR);
   COLOUR:=(MONITOR='INCOL');
END;

(****************************************************************)
PROCEDURE PREPAREGRAPHICS;
(****************************************************************)
BEGIN
   INITSCREEN;(* sets up graphics screen but leaves monitor in text mode*)
   DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE2);
   DRAWAXES(XCON,YCON,100,WHITE2);
   WSTAT(XCON+10,150,'pH vs titrant');
END;

(****************************************************************)
PROCEDURE FIN;
(****************************************************************)
VAR  X,Y: INTEGER;
BEGIN
  PAGE(OUTPUT);
  X:=5; Y:=6;
  WRITE(AT(X+2,Y),'REMOVE DISK FROM DISK DRIVE');  Y:=Y+5;
  WRITE(AT(X,Y),'IT WILL BE NECESSARY TO REBOOT'); Y:=Y+2;
  WRITE(AT(X,Y),'COMPUTER TO RUN ANOTHER PROGRAM    ');
  REPEAT
     X:=Y;   (* INFINITE LOOP*)
  UNTIL (X>Y);
END; (* FIN *)



BEGIN (* MAIN *)
  PREPAREGRAPHICS;
  CHECKCOL;
  SWAPGPON;  (* set swapping to level 2 *)
  REPEAT
     SHOWMENU(PROGNUM);
      IF NOT QUIT THEN INSTRUCT;
  UNTIL (QUIT OR READY);
  IF QUIT THEN FIN  ELSE  CHAINTO(PROGNUM);
END. (*MENU*)
```

```
(*$S++*)(*$R-*)(*$V-*)
(* Simulated titration between - strong acid/strong base, strong acid/weak base
                                  - weak acid/strong base, diprotic acid/strong base*)
PROGRAM TITRATION;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,TITRLIB;
CONST
   XCON=160; YCON=40;                                        (*origin of ph graph *)
   VOLSCALE=100;                         (*no. pixels on horizontal axis of ph graph *)
TYPE
   REALPTS=ARRAY[1..VOLSCALE] OF REAL;
   INTPTS=ARRAY[0..VOLSCALE] OF INTEGER;
VAR
   NEWIND,                                       (* selection of a new indicator      *)
   COLOUR,                                       (* is a colour monitor available     *)
   AGAIN : BOOLEAN;                               (* option to repeat titration        *)
   VOLPTS: REALPTS;                          (*vol.of titrant used initially to plot curve*)
   PHPTS:  INTPTS;      (*pH values corresponding to volpts required to plot entire curve -
                   these integer values have been scaled for graph by a factor of phratio*)
   INDNUM: CHAR;                                            (*   indicator number     *)

(**************************************************************)
PROCEDURE TITRATE;
(**************************************************************)
VAR
   ACIDCOL,BASECOL,MIDCOL,                            (*soln colour during titration *)
   SOLNCOL : SCREENCOLOR;                           (*current colour of soln in flask *)
   FIRSTDR,                        (*flag set at beginning of titration and atendpt to indicate
                                              that next drop will require a change in label*)
   LABELS,                                    (* option to show "state" of soln in flask*)
   SPACEPR,                                    (* flag to indicate space bar pressed     *)
   SELECTCHANGE   : boolean;       (*flag to indicate change intitrant increment volume *)
   INCR,                                      (* current titrant increment vol. selected *)
   BURVOL,                                    (* total vol. added from burette (titrant) *)
   ACIDVOL,BASEVOL,                             (* total vol. of acid & base in flask    *)
   ENDPT1,                                    (* vol. of titrant required to reach endpt *)
   PH,                                                   (* current pH of solution      *)
   NEXTVOL,                                        (* nextvol required for graphing    *)
   XTRAVOL,                                     (* vol. of titrant added not yet shown to
                                                     fill flask-only vol.>=5ml shown  *)
   PHRATIO : REAL;                     (* ratio between no. pixels on pHscale& pH range to
                                                               be plotted on scale    *)
   INTENDPT,                                   (* integer value of the endpt.-required for
                                                            indicator colour change    *)
   RTSIDE,LTSIDE,                            (* current x-coords. of flask being filled *)
   OLDLEVEL,INCREASE,  (* current & increase in level of soln  required for filling flask*)
   OLDX,OLDY,                                         (* current coord. of pH graph    *)
   INDEX :INTEGER;                                 (* required for graphing pH curve    *)
   NEWEQ: SHORTSTR;                                (* label of "state" of soln in flask    *)
```

```
(*=================================================================*)
PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
(*=================================================================*)
CONST  PHSCALE=100.0;  (* No. pixels on pH scale*)
       PHRANGE=14.0;   (* pH 0 - 14 equally spaced on pHscale*)
VAR    VOLSTR: SHORTSTR;
       X,Y:INTEGER;
BEGIN
  X:=XCON+8;
  Y:=YCON+110;
  IF COLOUR THEN
    BEGIN
      ACIDCOL:=VIOLET; BASECOL:=BLUE; MIDCOL:=GREEN;
    END
    ELSE
     BEGIN  (* not colour monitor *)
       ACIDCOL:=WHITE1;
       IF TITRTYPE=DIPROTIC THEN
       BEGIN
         MIDCOL:=BLACK1;  BASECOL:=ORANGE;
       END
        ELSE BASECOL:=BLACK1;
     END;

  BURVOL:=0.0;
  IF INFLASK=ACID THEN
   BEGIN
     ACIDVOL:=FLASKVOL;
     BASEVOL:=BURVOL;
     ENDPT1:=ACIDVOL*HCONC/OHCONC;
     SOLNCOL:=ACIDCOL;
     WSTAT(X,Y,'pH vs vol.base');
   END
   ELSE
   BEGIN
     ACIDVOL:=BURVOL;
     BASEVOL:=FLASKVOL;
     ENDPT1:=BASEVOL*OHCONC/HCONC;
     SOLNCOL:=BASECOL;
     WSTAT(X,Y,'pH vs vol.acid');
   END;

   IF ENDPT1<320.0 THEN INTENDPT:=ROUND(ENDPT1*100)
     ELSE INTENDPT:=32000; (* intendpt is used to changeindicator and change labels
       - flask will not hold 300mL therefore do not worry about endpts > 320. *)
   PHRATIO:=(PHSCALE/PHRANGE); (* pH increm. per pixel *)
   FILLRATE:=2;  (*determines rate at which flask filled*)
   REALSTR(ACIDVOL,VOLSTR,2,6);
   ACIDISP(VOLSTR);          (* Display acid volume*)
   REALSTR(BASEVOL,VOLSTR,2,6);
   BASEDISP(VOLSTR);          (* Display base volume *)
 END; (*INITCONDITIONS*)
```

```
(*==================================================================*)
PROCEDURE INITGRAPH;
(*==================================================================*)
BEGIN
 OLDX:=XCON;OLDY:=PHPTS[0]+YCON;
  INDEX:=0; NEXTVOL:=VOLPTS[1];
END;

(*==================================================================*)
PROCEDURE INITLEVEL;
(*==================================================================*)
(* initializes coord of sides of flask & top of flask as well as level of soln in flask   *)
CONST WIDTH=2; (* indent soln from sides of flask *)
BEGIN
   FLASKTOP:=FLASKY+ (3*FLASKSIZ)DIV 4; (*y-coord. of top sloping sides of flask*)
   NECKTOP:=FLASKY+FLASKSIZE;   (*y-coord. of very top of flask *)
   LTSIDE:=FLASKX-(FLASKSIZ DIV 2)+WIDTH; (*calc. coord of sides*)
   RTSIDE:=FLASKX+(FLASKSIZ DIV 2)-WIDTH; (*of flask given midpt. of base*)
   OLDLEVEL:=FLASKY+1;              (* base of flask= Flasky  *)
   INCREASE:=10;          (*depth of soln to be initially placed in flask*)
   XTRAVOL:=0.0;             (*initialize increment in titrant *)
END;(*INITLEVEL*)

(*==================================================================*)
PROCEDURE UPDATEQ;
(*==================================================================*)
(* Update current string relating to titration*)
CONST   WACID='W.ACID'; WBASE='W.BASE'; SBASE=' BASE ';
        SACID=' ACID  '; ABUFFER=' BUFFER';EQUIV=' END PT';
        (* strings to be displayed at appropriate stages of titration*)
        X=40; Y=27;  (* coord. at which string displayed*)
VAR     INTVOL: INTEGER;
BEGIN
   INTVOL:=ROUND(BURVOL*100);
   FIRSTDR:=FALSE;
   IF BURVOL=0 THEN
    BEGIN
     CASE TITRTYPE OF
       WEAKACID: IF INFLASK=ACID THEN NEWEQ:=WACID  ELSE NEWEQ:=SBASE;
       WEAKBASE: IF INFLASK=ACID THEN NEWEQ:=SACID  ELSE NEWEQ:=WBASE;
       STRONGACID:IF INFLASK=ACID THEN NEWEQ:=SACID  ELSE NEWEQ:=SBASE;
       DIPROTIC: NEWEQ:='DIACID';
     END; (* CASE *)
     FIRSTDR:=TRUE;
   END
  ELSE
  BEGIN
    IF TITRTYPE=DIPROTIC THEN
      BEGIN
        IF INTVOL=INTENDPT THEN NEWEQ:=' ENDPT1'        ELSE
        IF (INTVOL DIV 2)=INTENDPT THEN NEWEQ:=' ENDPT2'   ELSE
        IF INTVOL<INTENDPT THEN NEWEQ:=ABUFFER    ELSE
        IF (INTVOL DIV 2)<INTENDPT THEN NEWEQ:=' 2SALTS' ELSE  NEWEQ:=SBASE;
        FIRSTDR:=((NEWEQ=' ENDPT1') OR (NEWEQ=' ENDPT2'));
      END
```

```
      ELSE  (* not diprotic *)
        BEGIN
          IF INTVOL=INTENDPT THEN
            BEGIN
              NEWEQ:=EQUIV;  FIRSTDR:=TRUE;
            END
            ELSE
            IF INTVOL<INTENDPT THEN
              BEGIN
                CASE TITRTYPE OF
                  WEAKACID:  IF INFLASK=ACID THEN NEWEQ:=ABUFFER;
                  WEAKBASE:  IF INFLASK=BASE THEN NEWEQ:=ABUFFER;
                 END;(*CASE*)
               END
            ELSE  (* Burvol>endpt*)
              BEGIN
                CASE TITRTYPE OF
                  WEAKACID: IF INFLASK=ACID
                              THEN NEWEQ:=SBASE ELSE NEWEQ:=ABUFFER;
                  WEAKBASE: IF INFLASK=BASE
                              THEN NEWEQ:=SACID ELSE NEWEQ:=ABUFFER;
                 STRONGACID:IF INFLASK=ACID
                              THEN NEWEQ:=SBASE ELSE NEWEQ:=SACID;
                END;(*CASE*)
              END; (*ELSE*)
      END;
    END;
    WSTAT(X,Y,'      ');  WSTAT(X,Y,NEWEQ);
  END; (*UPDATEQ*)

  (*=================================================================*)
  PROCEDURE SETUPARRAYS(VAR VOLPTS:REALPTS; VAR PHPTS: INTPTS);
  (*=================================================================*)
  (* calculate pH value for volscale no. points. Volume calculated is twice required to
     reach end point if  monoprotic  and three times if diprotic *)
  CONST  MIN=0.05; MAX=10.00;  (*min & max value of incr. of titrant *)
         X=195; Y=10;     (*coord . for display of increment selected*)
  VAR    I:INTEGER; CH:CHAR;
         VOLRATIO:REAL;  (*ratio of vol. of titrant plotted to no. pixels onx- axis*)
         PROMPT,INCSTR:STRING; PHSTR:SHORTSTR;

    (*---------------------------------------------------------------*)
    PROCEDURE INITARRAYS;
    (*---------------------------------------------------------------*)
    BEGIN
      PHPTS[0]:=ROUND(PH*PHRATIO);
      IF TITRTYPE=DIPROTIC THEN VOLRATIO:=(ENDPT1*3.0)/VOLSCALE
        ELSE  (* NOT DIPROTIC *)
          VOLRATIO:=(ENDPT1*2.0)/VOLSCALE;     (* vol.incr. for each pixel*)
      FOR I:=1 TO VOLSCALE DO VOLPTS[I]:=VOLRATIO*I; (*total vol.at point 'I'*);
    END; (* INITARRAYS  *)
```

```
(*--------------------------------------------------------------------*)
PROCEDURE INFORMPH;
(*--------------------------------------------------------------------*)
BEGIN
 WSTAT(3,10,CONCAT('Initial pH is ',PHSTR));
 WSTAT(65,0,'Press <SPACE BAR> to continue');
END; (*INFORMPH*)
(*--------------------------------------------------------------------*)
PROCEDURE LABELOPTION;
(*--------------------------------------------------------------------*)
BEGIN
  WSTAT(3,10,'Do you want solution');
  WSTAT(3,0,'in flask labelled?  (Y/N)');
END; (* LABELOPTION *)

(*--------------------------------------------------------------------*)
PROCEDURE PLEASEWAIT;
(*--------------------------------------------------------------------*)
BEGIN
     WSTAT(10,5,'PREPARING SOLUTIONS ......');
END;

(*--------------------------------------------------------------------*)
PROCEDURE CYCLE;
(*--------------------------------------------------------------------*)
BEGIN
  IF NOT AGAIN THEN
   WHILE ((I<VOLSCALE) AND (NOT KEYIN)) DO
     BEGIN
       I:=I+1;
       IF INFLASK=ACID THEN
       CALCPH(FLASKVOL,VOLPTS[I],HCONC,OHCONC,PH)
          ELSE  CALCPH(VOLPTS[I],FLASKVOL,HCONC,OHCONC,PH);
      PHPTS[I]:=ROUND(PHRATIO*PH);
     END;(* WHILE *)
END; (* CYCLE *)


BEGIN (*SETUPARRAYS*)
   CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
   REALSTR(PH,PHSTR,2,5);
   IF (NOT AGAIN) THEN INITARRAYS;
   I:=0;
   LABELOPTION;
   CYCLE;
   GETHICHAR(X,Y-10,CH,['Y','N','Q']);
   REMOVERESPONSE(X,Y-10,1);
   LABELS:=CH='Y';  QUIT:=CH='Q';
   CHARTYPE(6);LABELOPTION; CHARTYPE(10);
   IF LABELS THEN UPDATEQ;
   IF QUIT THEN EXIT(SETUPARRAYS);
  INCRPROMPT;
  CYCLE;
  INRANGERESPONSE(INCR,INCSTR,MIN,MAX,X,Y);  (*get increment*)
  CHARTYPE(6);INCRPROMPT;CHARTYPE(10);(*erase prompt for incr.*)
  IF QUIT THEN EXIT(SETUPARRAYS);
```

```
        INFORMPH;         (*Inform initial pH*)
        CYCLE;
        GETACHAR(CH,[SPACE,'Q']);
        CHARTYPE(6);INFORMPH;CHARTYPE(10);
        QUIT:=CH='Q';
        IF QUIT THEN EXIT(SETUPARRAYS);
        IF ((I<VOLSCALE) AND (NOT AGAIN))THEN
          BEGIN
            PLEASEWAIT;   (*Display prompt*)
            REPEAT
              CYCLE;
              IF KEYIN THEN READ(CH);
            UNTIL I=VOLSCALE;
            CHARTYPE(6);PLEASEWAIT;CHARTYPE(10); (*erase prompt*)
          END;
        DISPLAYPH(PHSTR); (* display initial pH*)
    END;(* SETUPARRAYS *)

(*$I :TITR2*)
```

```
(* Titr2.text - included in TITRATE *)
   (*=================================================================*)
   PROCEDURE ADDMORE;
   (*=================================================================*)
   (* Increment vol. of titrant & calc new pH; display new pH & vol. of titrant*)
   CONST BLANK='     ';
   VAR   VOL: INTEGER; PHSTR,VOLSTR:  SHORTSTR;
   BEGIN (* ADDMORE*)
    BURVOL:=BURVOL+INCR;            (* calculate total vol. of titrant   *)
    VOL:=ROUND(BURVOL*100);        (* this prevents build up of floating *)
    BURVOL:=VOL/100.0;                                (* point errors *)
    REALSTR(BURVOL,VOLSTR,2,6);           (* convert vol. to string   *)
    CASE INFLASK OF                  (* display vol of titrant on screen *)
     ACID: BEGIN                      (* as either vol. of base or acid   *)
             BASEVOL:=BURVOL;
             BASEDISP(BLANK);
             BASEDISP(VOLSTR);
           END;
     BASE: BEGIN
             ACIDVOL:=BURVOL;
             ACIDISP(BLANK);
             ACIDISP(VOLSTR);
           END;
     END; (* CASE *)
    CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
    DISPLAYPH(BLANK);                           (* Erase old pH      *)
    REALSTR(PH,PHSTR,2,5);                     (* convert to string *)
    DISPLAYPH(PHSTR);                          (* Display new pH    *)
   END;  (* ADDMORE *)

   (*=================================================================*)
   PROCEDURE CHECKINDICATOR;
   (*=================================================================*)
   (* check if volume of titrant added has reached or exceeded end point and if indicator
   has not yet changed colour then do so *)
   VAR  INTVOL: INTEGER;
      (*--------------------------------------------------------------*)
      PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
      (*--------------------------------------------------------------*)
      (*Change colour of soln to newcolor & change label of soln in flask*)
      VAR   CURRENTL,DEPTH : INTEGER;
      BEGIN
        SOLNCOL:=NEWCOLOR;
        CURRENTL:=OLDLEVEL;
        INITLEVEL;
        DEPTH:=CURRENTL-OLDLEVEL;
        FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
        IF CURRENTL>OLDLEVEL THEN
          BEGIN    (* if colour change with soln in neck of flask*)
            DEPTH:=CURRENTL-OLDLEVEL;
            FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
          END;
        IF LABELS THEN UPDATEQ;
      END; (*CHANGECOL*)
```

```
BEGIN (*CHECKINDICATOR*)
  INTVOL :=ROUND(BURVOL*100);
  IF TITRTYPE=DIPROTIC THEN
    BEGIN
      IF SOLNCOL=ACIDCOL THEN
        BEGIN
          IF (INTVOL DIV 2 >=INTENDPT) THEN CHANGECOL(BASECOL)
            ELSE IF (INTVOL>=INTENDPT) THEN CHANGECOL(MIDCOL);
        END
          ELSE   IF (SOLNCOL=MIDCOL) AND (INTVOL DIV 2 >=INTENDPT)
                THEN CHANGECOL(BASECOL);
    END (*DIPROTIC*)
    ELSE (* NOT DIPROTIC *)
    BEGIN
      CASE INFLASK OF
        ACID  : IF (SOLNCOL=ACIDCOL) AND (INTVOL>=INTENDPT)
                  THEN CHANGECOL(BASECOL);
        BASE  : IF (SOLNCOL=BASECOL) AND (INTVOL>=INTENDPT)
                  THEN CHANGECOL(ACIDCOL);
        END; (*CASE*)
    END;
 END; (* CHECKINDICATOR*)

(*===============================================================*)
PROCEDURE GRAPH(VAR OLDX,OLDY,INDEX : INTEGER;
                                  VAR NEXTVOL : REAL; COL : SCREENCOLOR);
(*===============================================================*)
(*To plot graph use values of pH already scaled & stored as  integers in PHLTS array.
Plot all points with  titrant volume of less than or equal to burette vol. *)
VAR  EXACTPH,           (*scaled integer pH calc.from current pH value*)
     X,Y : INTEGER;                        (*new x,y coord of graph *)
BEGIN
   MOVECOL(OLDX,OLDY,COL);               (* move to last point plotted *)
   WHILE (BURVOL>=NEXTVOL) AND (INDEX<VOLSCALE) DO
     BEGIN
       INDEX :=INDEX+1 ;
       X:=INDEX+XCON; Y :=PHPTS[INDEX]+YCON;
       MOVETO(X,Y);
       NEXTVOL :=VOLPTS[INDEX+1];
       OLDX :=X;
       OLDY :=Y;
     END;
     IF (INDEX<VOLSCALE) THEN
       BEGIN
         EXACTPH :=ROUND(PH*PHRATIO);
         MOVETO(OLDX,EXACTPH+YCON);
         OLDY :=EXACTPH+YCON;
       END;
   PENCOLOR(NONE);
 END; (*GRAPH*)
```

```
(*==================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*==================================================================*)
(*Volume of solution in flask is only shown to increase when a suitable volume
(say 5mL or more) has been released from burette. Therefore smaller increments are
summed until this volume is reached and then level of soln is shown to rise *)
VAR   EXTRA : INTEGER;
BEGIN
  XTRAVOL:=XTRAVOL+INCR; (*xtravol. is vol.titrant added that has not yet been
                                                    shown to fill flask*)
 IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled
                 by an extra amt.  This value must be even due to slope of flask*)
   BEGIN
     EXTRA:=TRUNC(XTRAVOL/FILLRATE);
     IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
     XTRAVOL:=0;
   END;
END; (*CHECKLEVEL*)


BEGIN   (* TITRATE *)
  INITCONDITIONS(ENDPT1);
  INITLEVEL;
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
  SETUPARRAYS(VOLPTS,PHPTS);
  INITGRAPH;
  IF NOT QUIT THEN
   BEGIN
     REQUEST;                        (*Display prompt to press space bar*)
     SELECTCHANGE:=FALSE;
     FIRSTDR:=TRUE;
       REPEAT
         CHECKKEY(SPACEPR,SELECTCHANGE);
         IF SPACEPR THEN                (*if space bar has been pressed *)
           BEGIN
             MOVEDROP(INCR,OLDLEVEL);
             ADDMORE;
             CHECKINDICATOR;
             IF ((FIRSTDR) AND (LABELS)) THEN UPDATEQ;
             CHECKLEVEL(XTRAVOL,INCR);
             GRAPH(OLDX,OLDY,INDEX,NEXTVOL,WHITE1);
           END;
         IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
      UNTIL QUIT;
     REQUEST;          (*erase prompt*)
     AGAIN:=(BURVOL>0); (* only give option to repeat 'again'
   END;                                    if titration has commenced*)
END; (* TITRATE *)
```

```
(*************************************************************)
PROCEDURE STARTAGAIN;
(*************************************************************)

  (*===========================================================*)
  PROCEDURE CHECKAGAIN;
  (*===========================================================*)
  CONST   X=0;                                (* coord. to enter input *)
  VAR CH :CHAR; Y :INTEGER;

      (*-------------------------------------------------------*)
      PROCEDURE KEEPCURVE;
      (*-------------------------------------------------------*)
       VAR  REPLY :CHAR;  Y1 :INTEGER;
       BEGIN
         Y1 :=10;
         PAGE(OUTPUT);
         WRITE(AT(X,Y1),'Do you wish to retain pH curve from');  Y1 :=Y1+2;
         WRITE(AT(X,Y1),'previous titration? (Y/N)');
         GETTEXTCHAR(X+28,Y1,REPLY,['Y','N','Q']);
         IF REPLY='N' THEN FILLBOX(150,265,32,164,BLACK1);
         QUIT :=(CH='Q');                          (* erase pH graph *)
       END; (*KEEPCURVE*)

   BEGIN (*CHECKAGAIN *)
    PAGE(OUTPUT);   Y :=7;
     WRITE(AT(X,Y),'Repeat previous titration       .....(R)');   Y :=Y+2;
     WRITE(AT(X,Y),'Select different titration      .....(S)');    Y :=Y+2;
     WRITE(AT(X,Y),'Quit - back to MAIN MENU        .....(Q)');  Y :=Y+3;
     WRITE(AT(X+10,Y),'SELECT  OPTION   ........(  )');
     GETTEXTCH(X+37,Y,CH,['R','S','Q']);
      AGAIN :=CH='R';
     QUIT :=CH='Q';   (* resets 'quit'  *)
     IF ((NOT QUIT) AND (NOT AGAIN)) THEN
      BEGIN
        KEEPCURVE;
         IF NOT QUIT THEN SELECTTYPE(TITRTYPE);
      END    ELSE FILLBOX(150,265,32,164,BLACK1); (*erase pH graph*)
   END; (* CHECKAGAIN*)
 BEGIN (* STARTAGAIN*)
   IF AGAIN THEN CHECKAGAIN ELSE SELECTTYPE(TITRTYPE);
   CLEARVALUES(AGAIN);
   PAGE(OUTPUT);
 END; (* STARTAGAIN *)
```

```
(*=================================================================*)
 PROCEDURE SETUPARRAYS(VAR VOLPTS:REALPTS; VAR PHPTS: INTPTS);
(*=================================================================*)
(*Calculate pH value for volscale no. pts.  Vol. calculated is twice required to reach end
pt if monoprotic  & three times if diprotic. *)
CONST   MIN=0.05; MAX=10.00;        (*min & max value of increment of titrant*)
        X=195;  Y=10;                  (*coord to dispaly increment selected *)
VAR  I:INTEGER;
     VOLRATIO:REAL;     (*ratio of vol. of titrant plotted to no. pixels on x axis*)
     INCSTR:STRING;
     CH:CHAR;
     PHSTR:SHORTSTR;

   (*--------------------------------------------------------------*)
   PROCEDURE INITARRAYS;
   (*--------------------------------------------------------------*)
   BEGIN
     PHPTS[0]:=ROUND(PH*PHRATIO);
     IF TITRTYPE=DIPROTIC THEN VOLRATIO:=(ENDPT1*3.0)/VOLSCALE
      ELSE  VOLRATIO:=(ENDPT1*2.0)/VOLSCALE; (*vol.increments for each pixel*)
     FOR I:=1 TO VOLSCALE DO VOLPTS[I]:=VOLRATIO*I; (*total vol. at point 'I'*);
   END; (* INITARRAYS  *)

   (*--------------------------------------------------------------*)
   PROCEDURE INFORMPH;
   (*--------------------------------------------------------------*)
    BEGIN
       WSTAT(3,10,CONCAT('Initial pH is ',PHSTR));
       WSTAT(65,0,'Press <SPACE BAR> to continue');
    END; (*INFORMPH*)

   (*--------------------------------------------------------------*)
    PROCEDURE PLEASEWAIT;
   (*--------------------------------------------------------------*)
    BEGIN
       WSTAT(10,5,'PREPARING SOLUTIONS . . . . . .');
    END;

   (*--------------------------------------------------------------*)
   PROCEDURE CYCLE;
   (*--------------------------------------------------------------*)
    BEGIN
      IF NOT AGAIN THEN WHILE ((I<VOLSCALE) AND (NOT KEYIN)) DO
        BEGIN
            I:=I+1;
            IF INFLASK=ACID THEN
               CALCPH(FLASKVOL,VOLPTS[I],HCONC,OHCONC,PH)
                 ELSE CALCPH(VOLPTS[I],FLASKVOL,HCONC,OHCONC,PH);
            PHPTS[I]:=ROUND(PHRATIO*PH);
        END;(* WHILE *)
    END; (* CYCLE *)
```

```
(*================================================================*)
PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
(*================================================================*)
CONST  PHSCALE=100.0;  (* no. pixels on pH scale*)
       PHRANGE=14.0;   (* pH 0 - 14 equally spaced on pHscale*)
VAR    X,Y:INTEGER;
       VOLSTR: SHORTSTR;
BEGIN
  X:=XCON+8;
  Y:=YCON+110;
  BURVOL:=0.0;
  IF INFLASK=ACID THEN
    BEGIN
      ACIDVOL:=FLASKVOL;
      BASEVOL:=BURVOL;
      ENDPT1:=ACIDVOL*HCONC/OHCONC;
      WSTAT(X,Y,'pH vs vol.base');
    END
  ELSE
    BEGIN
      ACIDVOL:=BURVOL;
      BASEVOL:=FLASKVOL;
      ENDPT1:=BASEVOL*OHCONC/HCONC;
      WSTAT(X,Y,'pH vs vol.acid');
    END;
  PHRATIO:=(PHSCALE/PHRANGE);    (* pH increm. per pixel *)
  FILLRATE:=2;           (*determines rate at which flask filled*)
  REALSTR(ACIDVOL,VOLSTR,2,6);
  ACIDISP(VOLSTR);             (* display acid volume*)
  REALSTR(BASEVOL,VOLSTR,2,6);
  BASEDISP(VOLSTR);          (* display base volume *)
  CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
END;  (*INITCONDITIONS*)

(*================================================================*)
PROCEDURE INITGRAPH;
(*================================================================*)
BEGIN
  OLDX:=XCON;OLDY:=PHPTS[0]+YCON;
  INDEX:=0; NEXTVOL:=VOLPTS[1];
END;

(*================================================================*)
PROCEDURE INITLEVEL(VAR RIGHTSID,LEFTSID,TOPLEVEL:INTEGER);
(*================================================================*)
(* initializes coord of sides of flask & top of flask as well as level of soln in flask*)
CONST  WIDTH=2;
BEGIN
  FLASKTOP:=FLASKY+(3*FLASKSIZ)DIV 4; (*y-coord of top sloping sides of flask*)
  NECKTOP:=FLASKY+FLASKSIZE;              (*y-coord of very top of flask*)
  LEFTSID:=FLASKX-(FLASKSIZ DIV 2)+WIDTH;        (*calc. coord of sides *)
  RIGHTSID:=FLASKX+(FLASKSIZ DIV 2)-WIDTH; (*flask given midpt of base*)
  TOPLEVEL:=FLASKY+1;                               (*base of flask-flasky*)
  INCREASE:=2*ROUND(FLASKVOL/10);   (*depth of soln to be placed in flask*)
  XTRAVOL:=0.0; (*initialize increment in titrant*)
END;(*INITLEVEL*)
```

```
(*$S++*)(*$R-*)(*$V-*)
(* Simulated titration between - strong acid/strong base, strong acid/weak base
                               - weak acid/strong base,  diprotic acid/strong base
User  selects an indicator to be used with each titration.  Simulation illustrates effects of
using different  indicators*)

PROGRAM INDICATOR;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,TITRLIB;
CONST
   XCON=160; YCON=40;                                         (*origin of ph graph *)
   VOLSCALE=100;                     (*no. pixels on horizontal axis of ph graph *)
TYPE
  REALPTS=ARRAY[1..VOLSCALE] OF REAL;
  INTPTS=ARRAY[0..VOLSCALE] OF INTEGER;
VAR
  NEWIND,                                    (* selection of a new indicator      *)
  COLOUR,                                    (* is a colour monitor available     *)
  AGAIN : BOOLEAN;                           (* option to repeat titration        *)
  VOLPTS: REALPTS;                      (*vol.of titrant used initially to plot curve*)
  PHPTS:  INTPTS;       (*pH values corresponding to volpts required to plot entire curve
                 - these integer values have been scaled for graph by factor of phratio*)
  INDNUM: CHAR;                                         (*   indicator number     *)

(***********************************************************************)
PROCEDURE TITRATE
(***********************************************************************)
VAR
  ACIDCOL,BASECOL,                                  (* soln colour during titration *)
  NEXTCOL,         (*due to problems of overwriting one colour on another, colour of pH
                     graph pen may have to change if it overlaps particular colours.*)
  GRAPHCOL,                                  (*current colour of pH graph pen   *)
  SOLNCOL : SCREENCOLOR;                      (*current colour of soln in flask  *)
  PENCHANGE,                               (* flag to indicate change of graphing pen
                                                  colours is necessary            *)
  TINT,            (* flag to indicate at pH range where indicator changing colour *)
  INDICATOR,                        (* flag to indicate change indicator colour    *)
  SPACEPR,                            (* flag to indicate space bar pressed         *)
  SELECTCHANGE : boolean;                  (* flag to indchange in titrant vol.  *)
  INCR,                                    (* current titrant increment vol.     *)
  BURVOL,                                  (* total vol. added from burette       *)
  ACIDVOL,BASEVOL,                         (* total vol. of acid & base in flask *)
  ENDPT1,                              (* vol. of titrant required to reach endpt*)
  PH,                                          (* current pH of soln             *)
  NEXTVOL,                                 (* next vol required for graphing     *)
  XTRAVOL :REAL;                (* vol. of titrant added but not yet shown to fill flask *)

  RTSIDE,LTSIDE,                         (* current x-coord. of flask being filled*)
  OLDLEVEL,INCREASE,                     (* current & increase in level of soln   *)
  OLDX,OLDY,                               (* current coord of pH graph           *)
  INDEX,                                 (* required for graphing pH curve        *)

  UPPER,LOWER: INTEGER;                   (*pH limits for indicator changing col. *)
  UPPERPH,LOWERPH,                                 (*     "       "       "        *)
  PHRATIO:REAL;
```

```
BEGIN    (* main *)
   SETCHAIN(':MENU');
   AGAIN:=FALSE;
   QUIT:=FALSE;
   INITSCREEN;
   SETCOLOUR;
   SELECTTYPE(TITRTYPE);
   WHILE (NOT QUIT) DO
     BEGIN
        DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
        DRAWAXES(XCON,YCON,VOLSCALE,WHITE1);
        GRAFMODE;
        IF ((NOT AGAIN) AND (NOT QUIT)) THEN
              SETUPCONDITIONS(HCONC,OHCONC,INFLASK);
        IF NOT QUIT THEN TITRATE;
        TEXTMODE;
        STARTAGAIN;
      END; (* while *)
   BACKTOMENU;
END. (*TITRATE*)
```

```
BEGIN(*SETUPARRAYS*)
   REALSTR(PH,PHSTR,2,5); (* determine pHstr of initial pH*)
   IF (NOT AGAIN) THEN INITARRAYS;
   I:=0;
   INCRPROMPT;
   CYCLE;
   INRANGERESPONSE(INCR,INCSTR,MIN,MAX,X,Y);  (* get increment*)
   INCRPROMPT;     (*erase prompt *)
   IF QUIT THEN EXIT(SETUPARRAYS);
    INFORMPH;        (*display initial pH*)
    CYCLE;
    GETACHAR(CH,[SPACE,'Q']);
    INFORMPH;
    QUIT:=CH='Q';
    IF QUIT THEN EXIT(SETUPARRAYS);
    IF ((I<VOLSCALE) AND (NOT AGAIN))THEN
      BEGIN
        PLEASEWAIT;   (*display prompt*)
        REPEAT
            CYCLE;
            IF KEYIN THEN READ(CH);
        UNTIL I=VOLSCALE;
       PLEASEWAIT;
      END;
    DISPLAYPH(PHSTR);
END;(* SETUPARRAYS *)

(*==============================================================*)
PROCEDURE SLIGHTCHANGE(CURRENTL:INTEGER);
(*==============================================================*)
(* slight traces of other colour*)
VAR  Y,RSIDE,LSIDE,SHADE,DEGREE:INTEGER;
      BITCOL:SCREENCOLOR;

    (*----------------------------------------------------------*)
    PROCEDURE DRAWDOTS(START,FIN,YY:INTEGER);
    (*----------------------------------------------------------*)
    VAR LENGTH :INTEGER;
    BEGIN
        LENGTH:=3;
        FIN:=FIN-LENGTH;
        START:=START+SHADE;
        WHILE START<FIN DO
          BEGIN
             MOVECOL(START,YY,BITCOL);
             START:=START+LENGTH;
             MOVECOL(START,YY,NONE);
             START:=START+DEGREE;
          END;
     END; (*DRAWDOTS*)

BEGIN (*SLIGHTCHANGE*)
  INITLEVEL(RSIDE,LSIDE,Y);
  IF SOLNCOL=ACIDCOL THEN
    BEGIN
      BITCOL:=BASECOL;
```

```
      DEGREE:=ROUND((UPPERPH-PH)/(UPPERPH-LOWERPH)*11);
     END
      ELSE
      BEGIN
        BITCOL:=ACIDCOL;
        DEGREE:=ROUND((PH-LOWERPH)/(UPPERPH-LOWERPH)*11);
      END;
  IF INDNUM='3' THEN DEGREE:=DEGREE+2;
  SHADE:=(DEGREE+1) DIV 3;
  IF SHADE=0 THEN SHADE:=1;
  CASE DEGREE OF
   1,5,7: Y:=Y+1;
   2,3  : LSIDE:=LSIDE+2;
   6,9  : BEGIN
           Y:=Y+3;
           RSIDE:=RSIDE-1;
           LSIDE:=LSIDE+4;
          END;
   END;(*CASE*)
  IF Y<FLASKTOP THEN
    REPEAT
       DRAWDOTS(LSIDE,RSIDE,Y);
       Y:=Y+2*SHADE;
       RSIDE:=RSIDE-SHADE;
       LSIDE:=LSIDE+SHADE;
    UNTIL(Y>=CURRENTL-1)OR(Y>=FLASKTOP); (* Y>=FLASKTOP *)
  WHILE (Y<NECKTOP) AND (Y<CURRENTL-1) DO
   BEGIN
     DRAWDOTS(LSIDE,RSIDE,Y);
     Y:=Y+2*SHADE;
   END;
  TINT:=FALSE;
 END; (*SLIGHTCHANGE*)

(*==============================================================*)
PROCEDURE ADDMORE;
(*==============================================================*)
(* increment vol. of titrant; calc. new pH; display new pH & new vol. of titrant*)
CONST BLANK='      ';
VAR  VOL: INTEGER;
     PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
   BURVOL:=BURVOL+INCR;
   VOL:=ROUND(BURVOL*100);
   BURVOL:=VOL/100.0;
   REALSTR(BURVOL,VOLSTR,2,6);
   CASE INFLASK OF
    ACID: BEGIN
            BASEVOL:=BURVOL;
            BASEDISP(BLANK);
            BASEDISP(VOLSTR);
           END;
    BASE: BEGIN
            ACIDVOL:=BURVOL;
            ACIDISP(BLANK);
            ACIDISP(VOLSTR);
```

```
        END;
    END; (* CASE *)

    CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
    DISPLAYPH(BLANK);          (* erase old pH *)
    REALSTR(PH,PHSTR,2,5);  (* display new pH *)
    DISPLAYPH(PHSTR);
  END;  (* ADDMORE *)

(*=============================================================*)
PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
(*=============================================================*)
(*Change colour of soln & change label of soln in flask*)
VAR  CURRENTL,DEPTH : INTEGER;
BEGIN
   SOLNCOL:=NEWCOLOR;
   CURRENTL:=OLDLEVEL;
   INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
   DEPTH:=CURRENTL-OLDLEVEL;
   FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
   IF CURRENTL>OLDLEVEL THE (*  colour change with soln in neck of flask*)
     BEGIN
      DEPTH:=CURRENTL-OLDLEVEL;
      FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
     END;
END;(*CHANGECOL*)

(*=============================================================*)
 PROCEDURE CHECKINDICATOR;
(*=============================================================*)
VAR TEMPCOL: SCREENCOLOR;
BEGIN
   IF INFLASK=ACID THEN
     BEGIN
        IF ((SOLNCOL=ACIDCOL) AND (PH>=LOWERPH)) THEN
          BEGIN
             IF PH>=UPPERPH THEN
                BEGIN
                   SOLNCOL:=BASECOL;
                   INDICATOR:=TRUE;
                END  ELSE  TINT:=TRUE;
          END;
     END
     ELSE
     BEGIN
        IF ((SOLNCOL=BASECOL) AND (PH<=UPPERPH)) THEN
           BEGIN
              IF PH<=LOWERPH THEN
                 BEGIN
                   SOLNCOL:=ACIDCOL;
                   INDICATOR:=TRUE;
                 END ELSE TINT:=TRUE;
           END;
     END;(*ELSE*)

   IF INDICATOR THEN
```

```
        BEGIN
           IF ((SOLNCOL=BLACK2) OR (SOLNCOL=BLUE) OR (SOLNCOL=ORANGE))
            THEN TEMPCOL:=WHITE  ELSE TEMPCOL:=WHITE1;
            IF GRAPHCOL<>TEMPCOL THEN
              BEGIN
                 NEXTCOL:=TEMPCOL;
                 PENCHANGE:=TRUE;
              END;
        END;
     END; (* CHECKINDICATOR*)

    (*================================================================*)
    PROCEDURE GRAPH(VAR OLDX,OLDY,INDEX : INTEGER;
    (*================================================================*)
    (*To plot graph use values of pH already scaled & stored as integers in pHpts array.
    Plot all pts with corresponding titrant vol. of less than or equal to burette volume*)
    VAR  EXACTPH,          (*scaled integer pH calculated from current pH value *)
          X,Y : INTEGER;      (*new x,y coord of graph *)

       (*--------------------------------------------------------------*)
       PROCEDURE SWAPPEN(NEWY:INTEGER);
       (*--------------------------------------------------------------*)
       BEGIN
          IF ((INFLASK=ACID) AND (NEWY>=LOWER)) OR
            ((INFLASK=BASE) AND (NEWY<=UPPER)) THEN
             BEGIN
                IF ((INFLASK=ACID) AND (OLDY<LOWER))THEN MOVETO(OLDX,LOWER)
                  ELSE
                     IF((INFLASK=BASE) AND (OLDY>UPPER)) THEN MOVETO(OLDX,UPPER);
                GRAPHCOL:=NEXTCOL;
                PENCOLOR(GRAPHCOL);
                PENCHANGE:=FALSE;
             END;
       END; (*SWAPPEN*)

    BEGIN (*GRAPH*)
      MOVECOL(OLDX,OLDY,GRAPHCOL); (*move to last point plotted*)
      WHILE (BURVOL>=NEXTVOL) AND (INDEX<VOLSCALE) DO
       BEGIN
        INDEX:=INDEX+1;
        X:=XCON+INDEX; Y:=PHPTS[INDEX]+YCON;
        IF PENCHANGE THEN SWAPPEN(Y);
        MOVETO(X,Y);
        NEXTVOL:=VOLPTS[INDEX+1];
        OLDX:=X;
        OLDY:=Y;
       END;
      IF (INDEX<VOLSCALE) THEN
        BEGIN
           EXACTPH:=ROUND(PH*PHRATIO)+YCON;
           IF PENCHANGE THEN SWAPPEN(EXACTPH);
           MOVETO(OLDX,EXACTPH);
           OLDY:=EXACTPH;
        END;
      PENCOLOR(NONE);
    END;(*GRAPH*)                                    (*$I :2INDICATOR*)
```

```
(* 2INDICATOR is included in INDICATOR *)
    (*===============================================================*)
    PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
    (*===============================================================*)
    VAR EXTRA : INTEGER;
    BEGIN
       XTRAVOL:=XTRAVOL+INCR; (*xtravol is vol.titrant added that has not yet
                                                    been shown to fill flask*)
     IF (XTRAVOL>=5.0) THEN (*When xtravol is sufficiently large then flask is filled by
                         an extra amount. This amt must be even due to slope of flask*)
       BEGIN
           EXTRA:=TRUNC(XTRAVOL/FILLRATE);
           IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
           FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
           XTRAVOL:=0;
       END;
     END;(*CHECKLEVEL*)

    (*===============================================================*)
    PROCEDURE INITIND(INDNUM:CHAR; VAR LOWERPH,UPPERPH: REAL;
                                          VAR ACIDCOL,BASECOL: SCREENCOLOR);
    (*===============================================================*)
     VAR INDSTR: STRING[14];

       (*------------------------------------------------------------*)
       PROCEDURE METHYLO;
       (*------------------------------------------------------------*)
       BEGIN
          LOWERPH:=3.1;UPPERPH:=4.4;
          ACIDCOL:=VIOLET;BASECOL:=ORANGE;
          INDSTR:='methyl orange';
       END;
       (*------------------------------------------------------------*)
       PROCEDURE METHYLR;
       (*------------------------------------------------------------*)
        BEGIN
           LOWERPH:=4.2;UPPERPH:=6.2;
           ACIDCOL:=VIOLET;BASECOL:=ORANGE;
           INDSTR:='methyl red';
        END;
       (*------------------------------------------------------------*)
       PROCEDURE LITMUS;
       (*------------------------------------------------------------*)
        BEGIN
          LOWERPH:=4.5;UPPERPH:=8.3;
           ACIDCOL:=VIOLET;BASECOL:=BLUE;
           INDSTR:='litmus';
         END;
        (*-----------------------------------------------------------*)
       PROCEDURE BROMOB;
        (*-----------------------------------------------------------*)
        BEGIN
          LOWERPH:=6.0;UPPERPH:=7.6;
           ACIDCOL:=ORANGE;BASECOL:=BLUE;
           INDSTR:='bromo.blue';
         END;
```

```
  (*----------------------------------------------------------------*)
  PROCEDURE PHENOL;
  (*----------------------------------------------------------------*)
  BEGIN
    LOWERPH:=8.3;UPPERPH:=10.0;
    ACIDCOL:=BLACK1;BASECOL:=VIOLET;
    INDSTR:='phenolphth.';
  END;
  (*----------------------------------------------------------------*)
  PROCEDURE THYMOL;
  (*----------------------------------------------------------------*)
  BEGIN
    LOWERPH:=9.3;UPPERPH:=10.6;
    ACIDCOL:=BLACK1;BASECOL:=BLUE;
    INDSTR:='thymolphth.';
  END;
  (*----------------------------------------------------------------*)
  PROCEDURE HYPOTH;
  (*----------------------------------------------------------------*)
  BEGIN
     IF INDNUM='8' THEN (*2ND end pt*)
        CALCPH(FLASKVOL,2*ENDPT1,HCONC,OHCONC,LOWERPH)
      ELSE
       IF INFLASK=ACID THEN
         CALCPH(FLASKVOL,ENDPT1,HCONC,OHCONC,LOWERPH)
          ELSE CALCPH(ENDPT1,FLASKVOL,HCONC,OHCONC,LOWERPH);
     UPPERPH:=LOWERPH;
     ACIDCOL:=ORANGE;BASECOL:=BLUE;
     INDSTR:='Hypothetical';
  END;(* HYPOTH *)

BEGIN (* INITIND *)
  CASE INDNUM OF
    '1': METHYLO;
    '2': METHYLR;
    '3': LITMUS;
    '4': BROMOB;
    '5': PHENOL;
    '6': THYMOL;
    '7','8': HYPOTH;
   END; (*CASE*)
  WSTAT(XCON-10,YCON-14,INDSTR);
  IF NOT COLOUR THEN
   BEGIN
     ACIDCOL:=BLACK1;
     BASECOL:=BLUE;
   END;
END; (* INITIND*)

(*================================================================*)
PROCEDURE SHOWRANGE(VAR LOWERPH,UPPERPH:REAL;
                            ACIDCOL,BASECOL:SCREENCOLOR);
(*================================================================*)
VAR  START,FIN : INTEGER;
```

```
(*-----------------------------------------------------------------*)
PROCEDURE MIDWAY;
(*-----------------------------------------------------------------*)
VAR CENTRY,LINE:INTEGER;

    PROCEDURE DRAWDOTS(X1,X2,Y:INTEGER; COL:SCREENCOL);
    VAR LENGTH,ON,OFF,INDENT:INTEGER;
    BEGIN
      CASE LINE OF
                0: BEGIN ON:=1; OFF:=2; END;
              1,3: BEGIN ON:=1; OFF:=3; END;
          2,4,5,6: BEGIN ON:=2; OFF:=2; END;
            7,8,9,
         10,11,12: BEGIN ON:=3; OFF:=2; END;
         13,14,15: BEGIN ON:=3; OFF:=1; END;
          END; (*CASE*)
     CASE LINE OF
       3,8,10,12: INDENT:=0;
          0,2,4,6: INDENT:=1;
            1,7,9: INDENT:=2;
      5,11,13,15: INDENT:=3;
            END; (*CASE*)
     IF ((LINE=1) OR (LINE=3)) THEN
         BEGIN
            IF ((COL=BASECOL) AND (ACIDCOL<>BLACK1)) THEN
              COL:=ACIDCOL ELSE COL:=BASECOL;
         END;

       LENGTH:=2;
        X2:=X2-LENGTH;
        X1:=X1+INDENT*LENGTH;
        WHILE X1<X2 DO
           BEGIN
              MOVECOL(X1,Y,COL);
              X1:=X1+ON*LENGTH;
              MOVECOL(X1,Y,NONE);
              X1:=X1+OFF*LENGTH;
           END;
       END; (*DRAWDOTS*)

 BEGIN (*MIDWAY*)
    IF ACIDCOL=BLACK1 THEN CENTRY:=LOWER+1
      ELSE CENTRY:=(UPPER+LOWER+1) DIV 2;
    DRAWDOTS(START,FIN,CENTRY,BASECOL);
    LINE:=1;
    WHILE(CENTRY+2*LINE)<UPPER DO
       BEGIN
         DRAWDOTS(START,FIN,CENTRY+2*LINE,BASECOL);
         IF ACIDCOL<>BLACK1 THEN
            DRAWDOTS(START,FIN,CENTRY-2*LINE,ACIDCOL);
         LINE:=LINE+1;
       END
 END; (*MIDWAY*)
```

```
    BEGIN (* SHOWRANGE *)
       LOWER:=ROUND(PHRATIO*LOWERPH);
       UPPER:=ROUND(PHRATIO*UPPERPH);
       LOWER:=YCON+LOWER;
       UPPER:=YCON+UPPER;
       START:=XCON+2;
       FIN:=XCON+VOLSCALE;
       VIEWPORT(START,FIN,YCON+2,LOWER);
       FILLSCREEN(ACIDCOL);
       VIEWPORT(START,FIN,UPPER,YCON+100);
       FILLSCREEN(BASECOL);
       VIEWPORT(XMIN,XMAX,YMIN,YMAX);
       MIDWAY;
       DRAWLINE(START,LOWER,FIN,LOWER,WHITE1);
       DRAWLINE(START,UPPER,FIN,UPPER,WHITE1);
    END; (*SHOWRANGE*)

    (*==============================================================*)
    PROCEDURE INITCOLOURS;
    (*==============================================================*)
    (*determine initial colour of indicator in solution *)
    BEGIN
      IF PH<((LOWERPH+UPPERPH)/2) THEN SOLNCOL:=ACIDCOL  ELSE SOLNCOL:=BASECOL;
      TINT:=((PH<UPPERPH) AND (PH>LOWERPH));
      IF TINT THEN
        BEGIN
           IF INFLASK=ACID THEN SOLNCOL:=ACIDCOL  ELSE  SOLNCOL:=BASECOL;
        END;
      IF ((SOLNCOL=BLACK2) OR (SOLNCOL=BLUE) OR (SOLNCOL=ORANGE))
        THEN GRAPHCOL:=WHITE2  ELSE GRAPHCOL:=WHITE1;
      NEXTCOL:=GRAPHCOL;
    END; (*INITCOLOURS*)

BEGIN   (* TITRATE *)
 INITCONDITIONS(ENDPT1);
 INITIND(INDNUM,LOWERPH,UPPERPH,ACIDCOL,BASECOL);
 INITCOLOURS;
 INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
 SHOWRANGE(LOWERPH,UPPERPH,ACIDCOL,BASECOL);
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
 IF TINT THEN SLIGHTCHANGE(OLDLEVEL);
 SETUPARRAYS(VOLPTS,PHPTS);
 INITGRAPH;
 PENCHANGE:=FALSE;
 IF NOT QUIT THEN
   BEGIN
     REQUEST;           (*display prompt for space bar *)
     SELECTCHANGE:=FALSE;
       REPEAT
         CHECKKEY(SPACEPR,SELECTCHANGE);
         IF SPACEPR THEN
           BEGIN
             INDICATOR:=FALSE;
             MOVEDROP(INCR,OLDLEVEL);
             ADDMORE;
             CHECKINDICATOR;
```

```
                IF INDICATOR THEN CHANGECOL(SOLNCOL)
                     ELSE IF TINT THEN SLIGHTCHANGE(OLDLEVEL);
                CHECKLEVEL(XTRAVOL,INCR);
                GRAPH(OLDX,OLDY,INDEX,NEXTVOL);
             END;
           IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
       UNTIL QUIT;
     REQUEST;          (*erase prompt*)
     AGAIN:=(BURVOL>0);
   END;
END; (* TITRATE *)

(*************************************************************)
PROCEDURE GETINDICATOR(VAR INDNUM:CHAR);
(*************************************************************)
CONST X=0; DOTS='   .....(';
      EQSTR=' equiv.pt)';
VAR   NUM:CHAR; Y: INTEGER;
BEGIN
   Y:=0;
   PAGE(OUTPUT);
   WRITE(AT(X,Y),AROW(40,'*'));
   Y:=Y+2;WRITE(AT(X+7,Y),'INDICATORS AVAILABLE:');
   Y:=Y+2;WRITE(AT(X,Y),AROW(40,'*'));
   Y:=Y+2;WRITE(AT(X,Y),'Methyl orange (3.1-4.4)   ',DOTS,'1)');
   Y:=Y+2;WRITE(AT(X,Y),'Methyl red (4.2-6.2)      ',DOTS,'2)');
   Y:=Y+2;WRITE(AT(X,Y),'Litmus (4.5-8.3)          ',DOTS,'3)');
   Y:=Y+2;WRITE(AT(X,Y),'Bromothymol blue (6.0-7.6)',DOTS,'4)');
   Y:=Y+2;WRITE(AT(X,Y),'Phenolphthalein (8.3-10.0)',DOTS,'5)');
   Y:=Y+2;WRITE(AT(X,Y),'Thymolphthalein (9.3-10.6)',DOTS,'6)');
   Y:=Y+2;
   IF TITRTYPE=DIPROTIC THEN
    BEGIN
       WRITE(AT(X,Y),'Hypothetical(1st',EQSTR,DOTS,'7)');
        Y:=Y+2;WRITE(AT(X,Y),'Hypothetical(2nd',EQSTR,DOTS,'8)');
       NUM:='8';
    END
      ELSE
        BEGIN
            WRITE(AT(X,Y),'Hypothetical  (',EQSTR,DOTS,'7)');
            NUM:='7';
        END;
   Y:=Y+3;WRITE(AT(X+7,Y),'SELECT INDICATOR  ',DOTS,'  )');
   GETTEXTCHAR(37,Y,INDNUM,['1'..NUM,'Q']);
   QUIT:=INDNUM='Q';
   IF QUIT THEN AGAIN:=FALSE;
   PAGE(OUTPUT);
END; (*GETINDIC*)

(*************************************************************)
PROCEDURE STARTAGAIN;
(*************************************************************)

    (*==============================================*)
    PROCEDURE CHECKAGAIN;
    (*==============================================*)
```

```
    CONST  X=0;        (* coord. to enter input *)
           DOTS='   .....(';
    VAR   CH:CHAR;
          Y: INTEGER;
    BEGIN
      Y:=6; PAGE(OUTPUT);
      WRITE(AT(X,Y),'Repeat previous titration with ');Y:=Y+1;
      WRITE(AT(X+7,Y),'same indicator     ',DOTS,'R)');Y:=Y+1;
      WRITE(AT(X+7,Y),'different indicator',DOTS,'D)');Y:=Y+3;
      WRITE(AT(X,Y),'Select different titration',DOTS,'S)');Y:=Y+3;
      WRITE(AT(X,Y),'Quit - back to main menu  ',DOTS,'Q)'); Y:=Y+3;
      WRITE(AT(X+10,Y),'SELECT  OPTION ',DOTS,'  )');
      GETTEXTCH(X+37,Y,CH,['R','D','S','Q']);
      IF AGAIN THEN AGAIN:=((CH='R') OR (CH='D'));
      QUIT:=CH='Q';   (* resets 'quit'  *)
      NEWIND:=((CH='D') OR (CH='S')); QUIT:=CH='Q';
      PAGE(OUTPUT);
      IF ((NOT QUIT) AND (NOT AGAIN)) THEN SELECTTYPE(TITRTYPE);
     END; (* CHECKAGAIN*)

BEGIN
  PAGE(OUTPUT);
  FILLBOX(150,270,24,164,BLACK1); (* erase pH graph *)
  IF AGAIN THEN CHECKAGAIN
   ELSE
     BEGIN
       NEWIND:=TRUE;
       SELECTTYPE(TITRTYPE);
     END;
  CLEARVALUES(AGAIN);
 END; (* STARTAGAIN *)


BEGIN  (* main *)
    AGAIN:=FALSE;
    QUIT:=FALSE;
    NEWIND:=TRUE;
    INITSCREEN;
    SETCOLOUR;
    SELECTTYPE(TITRTYPE);
    WHILE (NOT QUIT) DO
      BEGIN
        DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
        DRAWAXES(XCON,YCON,VOLSCALE,WHITE1);
        IF ((NOT QUIT) AND (NEWIND)) THEN GETINDICATOR(INDNUM);
        GRAFMODE;
        IF ((NOT AGAIN) AND (NOT QUIT)) THEN
           SETUPCONDITIONS(HCONC,OHCONC,INFLASK);
        IF NOT QUIT THEN TITRATE;
        TEXTMODE;
        STARTAGAIN;
      END; (* while *)
    BACKTOMENU;
END. (*INDICATOR*)
```

```
(* QUIZ program is part of ACID/BASE TITRATION PACKAGE.  In QUIZ the computer selects
concentration of unknown solution, and user must carry out simulated titration in order to
determine this concentration.  The computer will inform the user as to the accuracy of
titration results.*)
(*$S++*) (*$R-*) (*$V-*) (*$I-*)

PROGRAM QUIZ;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,APPLESTUFF,USEFUL,TITRLIB;
VAR
  COLOUR,                                          (* is colour monitor available *)
  AGAIN : BOOLEAN;                                 (* option to repeat titration  *)
  UNKNOWN: ACIDORBASE;                             (* type of unknown solution    *)
  UNKNOWNC,                                        (* conc. of unknown solution   *)
  STDSOLN :REAL;                                   (* conc. of standard solution  *)
  SOLSTR: STRING[13];                              (* string of type of unknown   *)

(*************************************************************)
PROCEDURE GETCALCULATOR;
(*************************************************************)
CONST  TOP=10;
        INDENT=20;
VAR  Y: INTEGER;
     NUM1,NUM2: REAL;
     ESCAPE:BOOLEAN;
     OP:CHAR;

   (*==========================================================*)
   PROCEDURE LAYOUT;
   (*==========================================================*)
   CONST  STAR='*';
   VAR    X,Y:INTEGER;
   BEGIN
      PAGE(OUTPUT);
      X:=10; Y:=2;
      WRITE(AT(X,Y),AROW(21,STAR));      Y:=Y+2;
      WRITE(AT(X,Y),' C A L C U L A T O R'); Y:=Y+2;
      WRITE(AT(X,Y),AROW(21,STAR));
      WRITE(AT(X,23),'<Q> QUIT     <C> CLEAR');
      X:=4; Y:=TOP;
      WRITE(AT(X,Y),'+'); Y:=Y+2;
      WRITE(AT(X,Y),'-'); Y:=Y+2;
      WRITE(AT(X,Y),'X'); Y:=Y+2;
      WRITE(AT(X,Y),'/');
   END; (* LAYOUT *)

   (*==========================================================*)
   PROCEDURE ENTERNUM(Y:INTEGER; VAR NUM:REAL);
   (*==========================================================*)
   CONST  X=10;
   VAR  S:SHORTSTR;
   BEGIN
     WRITE(AT(X,Y),'ENTER NUM:');
     GETRESPONSE(22,Y,S,8,['0'..'9','.','Q','C']);
     WRITE(AT(X,Y),'          ');
     QUIT:=((S='Q') OR (S='C'));
     ESCAPE:=S='Q';
```

```
  NUM:=RVALUE(S);
 END; (* ENTERNUM *)

 (*==============================================================*)
 PROCEDURE ENTEROP(Y:INTEGER; VAR OP:CHAR);
 (*==============================================================*)
 BEGIN
  GOTOXY(INDENT,Y);
  GETACHAR(OP,['+','-','X','*','/','Q','C']);
  WRITE(OP);
  QUIT:=((OP='Q') OR (OP='C'));
  ESCAPE:=OP='Q';
 END;

 (*==============================================================*)
 PROCEDURE CLEAR(A,B: INTEGER);
 (*==============================================================*)
 CONST  BLANKL='         ';
 VAR   J: INTEGER;
 BEGIN
  FOR J:=A TO B DO WRITE(AT(INDENT,J),BLANKL);
 END; (* CLEAR*)

 (*==============================================================*)
 PROCEDURE CALC(VAR NUM1,NUM2:REAL;  OP:CHAR);
 (*==============================================================*)
 BEGIN
  CASE OP OF
      '+': NUM1:=NUM1+NUM2;
      '-': NUM1:=NUM1-NUM2;
  'X','*': NUM1:=NUM1*NUM2;
      '/': NUM1:=NUM1/NUM2;
    END; (*CASE*)

   WRITE(AT(INDENT,Y),NUM1:9:5);
  END; (* CALC *)

BEGIN (*CALCULATOR*)
  LAYOUT;
  REPEAT
     Y:=TOP;
     ENTERNUM(Y,NUM1);
     IF NOT QUIT THEN
        REPEAT
          Y:=Y+2;
          ENTEROP(Y,OP);
          IF NOT QUIT THEN
              BEGIN
                Y:=Y+2;
                ENTERNUM(Y,NUM2);
                CLEAR(TOP,Y);
                Y:=TOP;
              END;
          IF NOT QUIT THEN CALC(NUM1,NUM2,OP);
       UNTIL QUIT;
    CLEAR(TOP,Y);
```

```
  UNTIL ESCAPE;
  QUIT:=FALSE;
  PAGE(OUTPUT);
END; (*CALCULATOR*)

(***********************************************************)
PROCEDURE GETSPACEBAR;
(***********************************************************)
VAR CH:CHAR;
BEGIN
  WRITE(AT(23,23),'Press <SPACE BAR>');
  GETACHAR(CH,[SPACE,'Q']);
  QUIT:=CH='Q';
END;(* GETSPACEBAR*)

(***********************************************************)
PROCEDURE GETK(VAR K1,K2 :REAL);
(***********************************************************)
VAR  PROMPT1,PROMPT2:STRING;
     CH:CHAR;
BEGIN (*GETK*)
    CASE TITRTYPE OF
      WEAKACID : K1:=1.76E-5;
      WEAKBASE :K1:=1.79E-5;
      DIPROTIC : BEGIN
                   PROMPT1:='Indicator will change colour';
                   PROMPT2:='at 2nd end pt.   Press <SPACE BAR>';
                   TWOPROMPTS(PROMPT1,PROMPT2);
                   GETACHAR(CH,[SPACE,'Q']);
                   QUIT:=CH='Q';
                   TWOPROMPTS(PROMPT1,PROMPT2);
                   K1:=5.9E-2; K2:=6.4E-5;
                 END;
    END; (*CASE*)
END;(* GETK *)

(***********************************************************)
PROCEDURE SELECTUNKNOWN;
(***********************************************************)

   (*==========================================================*)
   PROCEDURE SELECTTYPE(VAR UNKNOWN:ACIDORBASE; VAR TITRTYPE:TITRAT);
   (*==========================================================*)
   CONST X=0;   ABLANK='             ...(';
   VAR  Y,K : INTEGER; CH:CHAR;
        S: ARRAY[1..5] OF STRING[18];
   BEGIN
     Y:=1;
     PAGE(OUTPUT);
     S[1]:='Hydrochloric acid';
     S[2]:='Sodium hydroxide ';
     S[3]:='Acetic acid      ';
     S[4]:='Ammonia          ';
     S[5]:='Oxalic acid      ';
     WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
     WRITE(AT(X,Y),'   TEST SOLUTIONS AVAILABLE:');
```

```
  Y:=Y+3;
  WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
  FOR K:=1 TO 5 DO
    BEGIN
      WRITE(AT(X,Y),S[K],ABLANK,K,')');
      Y:=Y+2;
    END;
  WRITE(AT(X,Y),'Quit - back to MAIN MENU         ...(Q)');
  Y:=Y+3;
  WRITE(AT(X,Y),'       SELECT SOLUTION   .........(  )');
  GETTEXTCHAR(37,Y,CH,['1'..'5','Q']);
  QUIT:=CH='Q';
  CASE CH OF
  'Q', '1': BEGIN
          TITRTYPE:=STRONGACID;
          SOLSTR:='Strong acid';
        END;
    '2': BEGIN
          TITRTYPE:=STRONGACID;
          SOLSTR:='Strong base';
        END;
    '3': BEGIN
          TITRTYPE:=WEAKACID;
          SOLSTR:='Weak acid';
        END;
    '4': BEGIN
          TITRTYPE:=WEAKBASE;
          SOLSTR:='Weak base';
        END;
    '5': BEGIN
          TITRTYPE:=DIPROTIC;
          SOLSTR:='Diprotic acid';
        END;
    END; (*CASE*)
  IF ((CH='2') OR (CH='4')) THEN UNKNOWN:=BASE ELSE UNKNOWN:=ACID;
  PAGE(OUTPUT);
END; (* SELECTYPE *)

(*=================================================================*)
 PROCEDURE GETUNKNOWN(VAR UNKNOWNC:REAL);
(*=================================================================*)
 VAR  MIN,MAX,NUM : INTEGER;

    (*-------------------------------------------------------------*)
    FUNCTION GETNUM(LOW,HIGH: INTEGER):INTEGER;
    (*-------------------------------------------------------------*)
    BEGIN
      GETNUM:=LOW + RANDOM MOD (HIGH-LOW+1);
    END; (* GETNUM *)

BEGIN
  (*$R APPLESTUFF*)
  RANDOMIZE;
  IF ODD(RANDOM) THEN
    BEGIN
      MIN:=1; MAX:=20;
```

```
        UNKNOWNC:=GETNUM(MIN,MAX)*5/100;
       END
       ELSE
        BEGIN
         MIN:=1; MAX:=9;
         UNKNOWNC:=GETNUM(MIN,MAX)/100;
        END;
     IF UNKNOWN=ACID THEN HCONC:=UNKNOWNC  ELSE OHCONC:=UNKNOWNC;
   END; (* GETUNKNOWN*)

BEGIN (* SELECTUNKNOWN*)
  SELECTTYPE(UNKNOWN,TITRTYPE);
  IF NOT QUIT THEN GETUNKNOWN(UNKNOWNC);
END; (* SELECTUNKNOWN *)

(*****************************************************************)
PROCEDURE STANDARDSOLN;
(*****************************************************************)
CONST STAR='*';
      X=0;  BLANK='       ';
VAR   S:STRING[20]; CH:CHAR;
      OK:BOOLEAN;
      Y,J,INTNUM : INTEGER;

   (*============================================================*)
   PROCEDURE GETSOLN(UNKN:REAL;VAR CONC:REAL);
   (*============================================================*)
   BEGIN
     IF TITRTYPE=DIPROTIC THEN UNKN:=2*UNKN;
     IF UNKN<=0.02 THEN CONC:=0.01 ELSE
     IF UNKN<0.08 THEN CONC:=0.05 ELSE
     IF UNKN<0.25 THEN CONC:=0.10 ELSE
     IF UNKN<0.75 THEN CONC:=0.50
     ELSE CONC:=1.00;
   END; (*GETSOLN*)

   (*============================================================*)
   PROCEDURE SHOWRANGE;
   (*============================================================*)
   BEGIN
     Y:=0;
     WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
     WRITE(AT(X,Y),BLANK,'UNKNOWN : ',SOLSTR);Y:=Y+2;
     WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+1;
     WRITE(AT(X,Y),'   The following standard solutions '); Y:=Y+1;
     WRITE(AT(X,Y),'are available:');Y:=Y+2;
     WRITE(AT(X,Y),BLANK,'0.010',S);Y:=Y+1;
     WRITE(AT(X,Y),BLANK,'0.050',S);Y:=Y+1;
     WRITE(AT(X,Y),BLANK,'0.100',S);Y:=Y+1;
     WRITE(AT(X,Y),BLANK,'0.500',S);Y:=Y+1;
     WRITE(AT(X,Y),BLANK,'1.000',S);
   END; (* SHOWRANGE *)
```

```
BEGIN  (* STANDARDSOLN *)
  IF UNKNOWN=BASE THEN S:='M Hydrochloric acid'  ELSE S:='M Sodium hydroxide';
  SHOWRANGE;
  Y:=16;
  WRITE(AT(X,Y),'Let COMPUTER select');      Y:=Y+1;
  WRITE(AT(X,Y),'appropriate standard solution   ....(C)');Y:=Y+2;
  WRITE(AT(X,Y),'YOU select standard solution    ....(Y)');Y:=Y+2;
  WRITE(AT(X,Y),'         SELECT OPTION     ........(  )');
  GETTEXTCHAR(37,Y,CH,['C','Y','Q']);
  QUIT:=CH='Q';
  IF QUIT THEN
    BEGIN
      PAGE(OUTPUT);
      EXIT(STANDARDSOLN);
    END;
  Y:=16;
  GOTOXY(X,Y);
  FOR J:=1 TO 6 DO WRITELN(BLANK,BLANK,BLANK,BLANK);
  IF CH='Y' THEN
    BEGIN
      WRITE(AT(X,Y),'Enter concentration');
      WRITE(AT(X,Y+1),'of standard solution:');
      REPEAT
        GETRESPONSE(24,Y+1,S,5,['1','0','.','5','Q']);
        QUIT:=S='Q';
        STDSOLN:=RVALUE(S);
        OK:=((STDSOLN>=0.01) AND (STDSOLN<=1.0));
        IF OK THEN
          BEGIN
             INTNUM:=TRUNC(STDSOLN*100);
             OK:=INTNUM IN [1,5,10,50,100];
          END;
        IF NOT OK THEN WRITE(AT(24,Y+1),BLANK);
      UNTIL OK OR QUIT;
    END
    ELSE
    BEGIN
      WRITE(AT(X,Y),AROW(40,STAR));
      GETSOLN(UNKNOWNC,STDSOLN);
      REALSTR(STDSOLN,S,3,5);
      WRITE(AT(X,Y+3),'STANDARD SOLUTION:   ',S,'M');
      WRITE(AT(X,Y+6),AROW(40,STAR));
      GETSPACEBAR;
    END;

    IF UNKNOWN=ACID THEN OHCONC:=STDSOLN  ELSE HCONC:=STDSOLN;
    PAGE(OUTPUT);
 END;  (* STANDARDSOLN *)

(*****************************************************************)
 PROCEDURE GETCONDITIONS;
(*****************************************************************)
 VAR ASTR: SHORTSTR;
   PROMPT1,PROMPT2:STRING;
```

```
(*=================================================================*)
PROCEDURE SELECTVOL(VAR FLASKVOL : REAL);
(*=================================================================*)
CONST  MIN=10.0; MAX=50;        (*  range of volume *)
       X=215; Y=10;             (*  coord. to enter input *)
BEGIN
  IF INFLASK=ACID THEN PROMPT1 :='acid' ELSE PROMPT1 :='base';
  PROMPT1 :=CONCAT('Select vol. of ',prompt1,' in flask:');
  PROMPT2:='  (10.0-50.0mL)';
  TWOPROMPTS(PROMPT1,PROMPT2);
  INRANGERESPONSE(FLASKVOL,ASTR,MIN,MAX,X,Y);
  TWOPROMPTS(PROMPT1,PROMPT2); (* erase *)
END; (*SELECTVOL*)

(*=================================================================*)
PROCEDURE SELECT(VAR INFLASK: ACIDORBASE);
(*=================================================================*)
VAR CH:CHAR;
BEGIN
   IF ((SOLSTR='Strong acid') OR (SOLSTR='Weak base')) THEN
      BEGIN
        PROMPT1 :='Basic';
        INFLASK:=BASE
      END
      ELSE
      BEGIN
        PROMPT1 :='Acidic';
        INFLASK:=ACID;
      END;
   PROMPT1 :=CONCAT(PROMPT1,' solution is in flask');
   PROMPT2:='        Press <SPACE BAR> to continue';
   TWOPROMPTS(PROMPT1,PROMPT2);
   GETACHAR(CH,[SPACE,'Q']);
   QUIT:=CH='Q';
   TWOPROMPS(PROMPT1,PROMPT2);  (* ERASE *)
END; (*SELECT*)

BEGIN (* GETCONDITIONS *)
 IF UNKNOWN=ACID THEN
  BEGIN
    REALSTR(OHCONC,ASTR,3,5);
    ACIDMOLARITY('?');
    BASEMOLARITY(ASTR);
  END
  ELSE
   BEGIN
     REALSTR(HCONC,ASTR,3,5);
     BASEMOLARITY('?');
     ACIDMOLARITY(ASTR);
   END;
  IF NOT QUIT THEN SELECT(INFLASK);
  IF NOT QUIT THEN SELECTVOL(FLASKVOL);
END;(* GETCONDITIONS *)
```

```
(*************************************************************)
PROCEDURE SHOWTYPE;
(*************************************************************)
CONST  S='Strong ';
VAR  X,Y: INTEGER;
      S1 : STRING[4];
BEGIN
   IF UNKNOWN=ACID THEN S1 :='Base' ELSE S1 :='Acid';
   X:=210; Y:=135;
   WSTAT(X-(7*LENGTH(SOLSTR) DIV 2),Y,SOLSTR); Y:=Y-20;
   WSTAT(X,Y,'VS'); Y:=Y-20;
   WSTAT(X-(7*5),Y,CONCAT(S,S1));
   IF INFLASK=ACID THEN S1 :='Base' ELSE S1 :='Acid'; Y:=Y-50;
   WSTAT(X-(7*8),Y,CONCAT(S1,' in burette'));
END; (* SHOWTYPE *)

(*$I :QUIZ2*)
```

```
(* QUIZ2 - INCLUDED IN QUIZ *)
(**************************************************************)
PROCEDURE TITRATE;
(**************************************************************)
VAR
 ACIDCOL,BASECOL,                           (* soln colour during titration   *)
 SOLNCOL : SCREENCOLOR;                     (*current colour of soln in flask *)
 SPACEPR,                                   (* flag to indicate space bar pressed*)
 SELECTCHANGE : boolean;        (* flag to indicate change in titrant increment vol.*)
 INCR,                                      (* current titrant increment volume  *)
 BURVOL,                                    (* total vol. added from burette     *)
 ACIDVOL,BASEVOL,                           (* total vol. of acid & base in flask*)
 ENDPT1,                                    (* vol. of titrant required to reach endpt*)
 PH,                                        (* currend pH of solution            *)
 XTRAVOL : REAL;               (* vol. of titrant added but not yet shown to fill flask *)
 INTENDPT,          (*integer value of endpt*100  required for indicator colour change*)
 RTSIDE,LTSIDE,                             (*current x-coord. of flask being filled*)
 OLDLEVEL,INCREASE,      (*current & increase in level of soln required to fill flask*)
 OLDX,OLDY,                                 (* current coord of pH graph          *)
 INDEX:INTEGER;                             (* required for graphing pH curve     *)

   (*============================================================*)
   PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
   (*============================================================*)
   CONST  X=160; Y=150;
   VAR   ASTR: SHORTSTR;
   BEGIN
    IF COLOUR THEN
      BEGIN ACIDCOL :=VIOLET; BASECOL :=BLUE; END
        ELSE  (* not colour monitor *)
           BEGIN  ACIDCOL :=WHITE1; BASECOL :=BLACK1; END;
    BURVOL :=0.0;
    IF INFLASK=ACID THEN
     BEGIN
       ACIDVOL :=FLASKVOL;
       BASEVOL :=BURVOL;
       ENDPT1 :=ACIDVOL*HCONC/OHCONC;
       SOLNCOL :=ACIDCOL;
     END
     ELSE
     BEGIN
       ACIDVOL :=BURVOL;
       BASEVOL :=FLASKVOL;
       ENDPT1 :=BASEVOL*OHCONC/HCONC;
       SOLNCOL :=BASECOL;
     END;
     IF ENDPT1<320.0 THEN INTENDPT :=ROUND(ENDPT1 * 100) ELSE INTENDPT :=32000;
     FILLRATE :=2;    (*determines rate at which flask filled *)
     REALSTR(ACIDVOL,ASTR,2,6);
     ACIDISP(ASTR);       (* display acid volume*)
     REALSTR(BASEVOL,ASTR,2,6);
     BASEDISP(ASTR);       (* display base volume *)
     CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
     REALSTR(PH,ASTR,2,5);
     DISPLAYPH(ASTR);
    END;  (*INITCONDITIONS*)
```

```
(*==============================================================*)
PROCEDURE INITLEVEL;
(*==============================================================*)
(*initializes coord of sides of flask & top of flask  as well as level of soln in flask*)
BEGIN
   FLASKTOP:=FLASKY+(3*FLASKSIZ)DIV 4; (*ycoord of top sloping sides of flask*)
   NECKTOP:=FLASKY+FLASKSIZE;   (*ycoord of very top of flask*)
   LTSIDE:=FLASKX-(FLASKSIZ DIV 2)+2;(*calc. coord of sides of*)
   RTSIDE:=FLASKX+(FLASKSIZ DIV 2)-2;(*flask given midpt of base*)
   OLDLEVEL:=FLASKY+1;             (* base of flask - flasky*)
   INCREASE:=2*ROUND(FLASKVOL/10);(*depth of soln to be initially placed in flask*)
   XTRAVOL:=0.0;(*initialize increment in titrant to be dispalyed*)
END;(*INITLEVEL*)

(*==============================================================*)
PROCEDURE ADDMORE;
(*==============================================================*)
(*increment volume of titrant & calculate new pH; display new pH & new vol. of
titrant*)
CONST BLANK='      ';
VAR VOL: INTEGER; PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
   BURVOL:=BURVOL+INCR;
   VOL:=ROUND(BURVOL*100);
   BURVOL:=VOL/100.0;
   REALSTR(BURVOL,VOLSTR,2,6);
   CASE INFLASK OF
        ACID: BEGIN
                  BASEVOL:=BURVOL;
                  BASEDISP(BLANK);
                  BASEDISP(VOLSTR);
                END;
       BASE: BEGIN
                  ACIDVOL:=BURVOL;
                  ACIDISP(BLANK);
                  ACIDISP(VOLSTR);
                END;
         END; (* CASE *)
   CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
   DISPLAYPH(BLANK);           (* erase old pH *)
   REALSTR(PH,PHSTR,2,5);  (*display new pH *)
   DISPLAYPH(PHSTR);
END;  (* ADDMORE *)
(*==============================================================*)
PROCEDURE CHECKINDICATOR;
(*==============================================================*)
VAR INTVOL:INTEGER;
```

```
(*---------------------------------------------------------------------*)
PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
(*---------------------------------------------------------------------*)
(*change colour of soln to newcolor *)
 VAR  CURRENTL,DEPTH  : INTEGER;
 BEGIN
     SOLNCOL:=NEWCOLOR;
     CURRENTL:=OLDLEVEL;
     INITLEVEL;
     DEPTH:=CURRENTL-OLDLEVEL;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
     IF CURRENTL>OLDLEVEL THEN
      BEGIN
        DEPTH:=CURRENTL-OLDLEVEL;
        FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
      END;
  END;(*CHANGECOL*)

BEGIN(*CHECKINDICATOR*)
   INTVOL:=ROUND(BURVOL * 100); (* NB. burvol <320mL *)
   IF TITRTYPE=DIPROTIC THEN
    BEGIN
      IF SOLNCOL=ACIDCOL THEN
       BEGIN
          IF (INTVOL DIV 2)>=INTENDPT THEN CHANGECOL(BASECOL);
       END;
    END (*DIPROTIC*)

    ELSE (* NOT DIPROTIC *)
     IF INTVOL>=INTENDPT THEN
       BEGIN
         CASE INFLASK OF
            ACID  : IF (SOLNCOL=ACIDCOL) THEN CHANGECOL(BASECOL);
            BASE  : IF (SOLNCOL=BASECOL) THEN CHANGECOL(ACIDCOL);
           END; (*CASE*)
       END;
 END; (* CHECKINDICATOR*)

(*=====================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*=====================================================================*)
VAR  EXTRA : INTEGER;
BEGIN
   XTRAVOL:=XTRAVOL+INCR; (*xtravol is vol. titrant added that has
                  not yet been shown to fill flask*)
   IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled by
                  an extra amt.  This amt must be even due to slope of flask*)
    BEGIN
      EXTRA:=TRUNC(XTRAVOL/FILLRATE);
      IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
      FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
      XTRAVOL:=0;
    END;
END; (*CHECKLEVEL*)
```

```
BEGIN   (* TITRATE *)
 INITCONDITIONS(ENDPT1);
 INITLEVEL;
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
 AGAIN:=TRUE;
 SELECTINCR(INCR);
 IF NOT QUIT THEN
  BEGIN
   REQUEST;          (*prompt to press space bar *)
   SELECTCHANGE:=FALSE;
    REPEAT
     CHECKKEY(SPACEPR,SELECTCHANGE);
     IF SPACEPR THEN
       BEGIN
        MOVEDROP(INCR,OLDLEVEL);
        ADDMORE;
        CHECKINDICATOR;
        CHECKLEVEL(XTRAVOL,INCR);
       END;
      IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
    UNTIL QUIT;
   REQUEST;          (*erase prompt *)
  END;
END; (* TITRATE *)

(*****************************************************************)
PROCEDURE STARTAGAIN;
(*****************************************************************)
CONST BLANK='    ';
      DOTS='......(';
      X=0;
VAR   CH:CHAR;  Y: INTEGER;
      UNSTR: SHORTSTR;

   (*===========================================================*)
   PROCEDURE CLEARVALUES;
   (*===========================================================*)
   (* erase all values and flask *)
   BEGIN
    DISPLAYPH(BLANK);
    BASEDISP('    ');
    ACIDISP('    ');
    FILLBOX(10,110,25,125,BLACK1);       (* erase flask  *)
    FILLBOX(150,265,32,164,BLACK1); (* erase pH graph *)
   END; (*CLEARVALUES*)

   (*===========================================================*)
   PROCEDURE GETGUESS;
   (*===========================================================*)
   VAR  S:STRING;
     CONC:REAL;
     PERCENT:INTEGER;
```

```
(*------------------------------------------------------------------*)
FUNCTION ERROR(NUM1,NUM2:REAL):INTEGER;
(*------------------------------------------------------------------*)
BEGIN
 ERROR:=TRUNC(100*(ABS(NUM1-NUM2)/NUM2));
END; (* ERROR *)

(*------------------------------------------------------------------*)
PROCEDURE SHOWRESULT(S:STRING);
(*------------------------------------------------------------------*)
VAR UNSTR:SHORTSTR;
BEGIN
 REALSTR(UNKNOWNC,UNSTR,3,5);
 Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,'*'));        Y:=Y+2;
 WRITE(AT(X,Y),BLANK,BLANK,S); Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,'*'));        Y:=Y+2;
 WRITE(AT(X,Y),'Concentration of unknown: ',unstr,'M');
 GETSPACEBAR;
 PAGE(OUTPUT);
END; (*SHOWRESULT*)

BEGIN (*GETGUESS*)
 PAGE(OUTPUT);
 Y:=6;
 WRITE(AT(X,Y),AROW(40,'*'));   Y:=Y+2;
 WRITE(AT(X,Y),BLANK,'UNKNOWN: ',SOLSTR);      Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,'*'));   Y:=Y+3;
 WRITE(AT(X,Y),' Calculated concentration: ');
 GETRESPONSE(30,Y,S,5,NUMS);
 CONC:=RVALUE(S);
 IF CONC<1.5 THEN
   PERCENT:=ERROR(CONC,UNKNOWNC)
    ELSE PERCENT:=50;(* error > 50 percent*)
 CASE PERCENT OF
  0 :   S:='EXCELLENT';
  1,2,3: S:='VERY CLOSE';
  4,5,6: S:='CLOSE';
  END; (*CASE*)
 IF PERCENT>=20 THEN S:='TERRIBLE'
  ELSE IF PERCENT>6 THEN S:='NOT TOO GOOD';
 SHOWRESULT(S);
END; (*GETGUESS*)

(*==================================================================*)
PROCEDURE ANOTHERGO;
(*==================================================================*)
CONST DOTS='    ......(';
BEGIN
  Y:=6;
  WRITE(AT(X,Y),'Another unknown solution',DOTS,'A)');Y:=Y+3;
  WRITE(AT(X,Y),'Quit back to MAIN MENU  ',DOTS,'Q)');Y:=Y+4;
  WRITE(AT(X+14,Y),'SELECT     .........(  )');
  GETTEXTCHAR(36,Y,CH,['A','Q']);
  QUIT:=CH='Q';
  IF NOT QUIT THEN SELECTUNKNOWN;
```

```
    PAGE(OUTPUT);
  END; (*ANOTHERGO*)

  (*=============================================================*)
  PROCEDURE GIVEUP;
  (*=============================================================*)
  VAR  UNSTR:SHORTSTR;
  BEGIN
   REALSTR(UNKNOWNC,UNSTR,3,5);
   WRITE(AT(0,10),'CONCENTRATION OF UNKNOWN WAS ',UNSTR,'M');
   GETSPACEBAR;
   PAGE(OUTPUT);
  END; (*GIVEUP*)

BEGIN (* STARTAGAIN *)
   Y:=6;
   WRITE(AT(X,Y),'REPEAT previous titration',BLANK,DOTS,'R)'); Y:=Y+2;
   WRITE(AT(X,Y),'Repeat titration');   Y:=Y+1;
   WRITE(AT(X,Y),'    but ALTER conditions ',BLANK,DOTS,'A)'); Y:=Y+2;
   WRITE(AT(X,Y),'ENTER concentration of unknown',DOTS,'E)'); Y:=Y+2;
   WRITE(AT(X,Y),'Get CALCULATOR ',BLANK,BLANK,BLANK,DOTS,'C)'); Y:=Y+2;
   WRITE(AT(X,Y),'GIVE up   ',BLANK,BLANK,BLANK,BLANK,DOTS,'G)');  Y:=Y+2;
   WRITE(AT(X,Y),'QUIT - back to MAIN MENU ',BLANK,DOTS,'Q)'); Y:=Y+3;
   WRITE(AT(X+10,Y),'SELECT  OPTION   ..',DOTS,'  )');
   GOTOXY(37,Y);
   GETTEXTCHAR(X+37,Y,CH,['R','A','C','E','G','Q']);
   IF AGAIN THEN AGAIN:=(CH='R') OR (CH='C');
   QUIT:=CH='Q';   (* resets 'quit'  *)
   PAGE(OUTPUT);
   CLEARVALUES(AGAIN);
   FILLBOX(150,265,32,164,BLACK1); (*erase right hand box*)
   CASE CH OF
      'E': BEGIN  GETGUESS;  ANOTHERGO;  END;
      'G': BEGIN  GIVEUP;    ANOTHERGO;  END;
      'C':BEGIN  GETCALCULATOR; STARTAGAIN; END;
     END; (* CASE *)
  PAGE(OUTPUT);
END; (* STARTAGAIN *)
```

```
BEGIN   (* main *)
(*$N+*)
(*$R TURTLEGRAPHICS,TRANSCEND,USEFUL,TITRLIB*)
    AGAIN:=FALSE;
    QUIT:=FALSE;
    INITSCREEN;
    SETCOLOUR;
    SELECTUNKNOWN;
    WHILE (NOT QUIT) DO
      BEGIN
         IF (NOT AGAIN) THEN STANDARDSOLN;
         DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
         GRAFMODE;
         IF (NOT AGAIN) THEN
          BEGIN
            GETK(K1,K2);
            IF NOT QUIT THEN GETCONDITIONS;
          END; (* if *)
         SHOWTYPE;
         IF NOT QUIT THEN TITRATE;
         TEXTMODE;
         STARTAGAIN;
      END; (* while *)
    BACKTOMENU;
    SETCHAIN(':MENU');
END. (*QUIZ*)
```

(* ASSIGNMENT is part of the ACID/BASE TITRATION PACKAGE. In ASSIGNMENT the user must enter an assignment no. which corresponds to a particular unknown solution. The user must then carry out a simulated titration in order to determine the concentration of this unknown. The program does not reveal the correct concentration of the unknown - the results must be submitted to the teacher for assessment.*)

```
(*$S++*)(*$R-*)(*$V-*)(*$I-*)

PROGRAM ASSIGNMENT;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,TITRLIB;
VAR
   COLOUR,                          (* is a colour monitor available     *)
   NEW,                             (* flag to select a new assignment   *)
   AGAIN : BOOLEAN;                 (* option to repeat titration        *)
   UNKNOWN: ACIDORBASE;             (* type of unknown soln              *)
   UNKNOWNC,                        (* concentration of unknown soln     *)
   STDSOLN :REAL;                   (* concentration of standard soln    *)
   SOLSTR: STRING[17];              (* string of type of unknown soln    *)


(****************************************************************)
PROCEDURE GETCALCULATOR;
(****************************************************************)
CONST TOP=10; INDENT=20;
VAR  Y: INTEGER;
     NUM1,NUM2: REAL;
     ESCAPE:BOOLEAN;
     OP:CHAR;

   (*==========================================================*)
   PROCEDURE LAYOUT;
   (*==========================================================*)
   CONST STAR='*';
   VAR X,Y:INTEGER;
   BEGIN
      PAGE(OUTPUT);
      X:=10; Y:=2;
      WRITE(AT(X,Y),AROW(21,STAR));      Y:=Y+2;
      WRITE(AT(X,Y),' C A L C U L A T O R'); Y:=Y+2;
      WRITE(AT(X,Y),AROW(21,STAR));
      WRITE(AT(X,23),'<Q> QUIT      <C> CLEAR');
      X:=4; Y:=TOP;
      WRITE(AT(X,Y),'+');Y:=Y+2;
      WRITE(AT(X,Y),'-');Y:=Y+2;
      WRITE(AT(X,Y),'X');Y:=Y+2;
      WRITE(AT(X,Y),'/');
   END; (* LAYOUT*)

  (*==========================================================*)
   PROCEDURE ENTERNUM(Y:INTEGER; VAR NUM:REAL);
   (*==========================================================*)
   CONST X=10;
   VAR S:SHORTSTR;
   BEGIN
      WRITE(AT(X,Y),'ENTER NUM:');
      GETRESPONSE(22,Y,S,8,['0'..'9','.','Q','C']);
      WRITE(AT(X,Y),'          ');
```

```
      QUIT :=((S='Q') OR (S='C'));
      ESCAPE :=S='Q';
      NUM :=RVALUE(S);
   END; (* ENTERNUM *)

   (*============================================================*)
   PROCEDURE ENTEROP(Y :INTEGER; VAR OP :CHAR);
   (*============================================================*)
   BEGIN
     GOTOXY(INDENT,Y);
     GETACHAR(OP,['+','-','X','*','/','Q','C']);
     WRITE(OP);
     QUIT :=((OP='Q') OR (OP='C'));
     ESCAPE :=OP='Q';
   END;

   (*============================================================*)
   PROCEDURE CLEAR(A,B : INTEGER);
   (*============================================================*)
   CONST  BLANKL='          ';
   VAR    J: INTEGER;
   BEGIN
    FOR J:=A TO B DO WRITE(AT(INDENT,J),BLANKL);
   END; (* CLEAR*)

   (*============================================================*)
   PROCEDURE CALC(VAR NUM1,NUM2 :REAL; OP :CHAR);
   (*=========================================================*)
   BEGIN
     CASE OP OF
       '+': NUM1 :=NUM1+NUM2;
       '-': NUM1 :=NUM1-NUM2;
   'X','*': NUM1 :=NUM1*NUM2;
      '/': NUM1 :=NUM1/NUM2;
      END; (*CASE*)

     WRITE(AT(INDENT,Y),NUM1 :9:5);
    END; (* CALC *)

BEGIN (*CALCULATOR*)
  LAYOUT;
  REPEAT
   Y:=TOP;
   ENTERNUM(Y,NUM1);
   IF NOT QUIT THEN
     REPEAT
      Y:=Y+2;
      ENTEROP(Y,OP);
      IF NOT QUIT THEN
        BEGIN
          Y:=Y+2;
          ENTERNUM(Y,NUM2);
          CLEAR(TOP,Y);
          Y:=TOP;
        END;
      IF NOT QUIT THEN CALC(NUM1,NUM2,OP);
```

```
    UNTIL QUIT;
   CLEAR(TOP,Y);
  UNTIL ESCAPE;
  QUIT:=FALSE;
  PAGE(OUTPUT);
END;

(*************************************************************)
PROCEDURE GETSPACEBAR;
(*************************************************************)
VAR CH:CHAR;
BEGIN
  WRITE(AT(23,23),'Press <SPACE BAR>');
  GETACHAR(CH,[SPACE,'Q']);
  QUIT:=CH='Q';
END;(* GETSPACEBAR*)

(*************************************************************)
PROCEDURE GETK(VAR K1,K2 :REAL);
(*************************************************************)
VAR  PROMPT1,PROMPT2: STRING;
     CH:CHAR;
BEGIN (*GETK*)
   CASE TITRTYPE OF
   WEAKACID  : K1:=1.76E-5;  (* Ka for acetic acid*)
   WEAKBASE  : K1:=1.79E-5;  (* Kb for ammonia*)
   DIPROTIC  : BEGIN
                  PROMPT1:='Indicator will change colour';
                  PROMPT2:='at 2nd end pt.   Press <SPACE BAR>';
                  TWOPROMPTS(PROMPT1,PROMPT2);
                  GETACHAR(CH,[SPACE,'Q']);
                  QUIT:=CH='Q';
                  CHARTYPE(6);
                  TWOPROMPTS(PROMPT1,PROMPT2);
                  CHARTYPE(10);
                  K1:=5.9E-2; K2:=6.4E-5;(* K values for oxalic acid*)
               END;
  END; (*CASE*)
END;(* GETK *)

(*************************************************************)
PROCEDURE SELECTUNKNOWN;
(*************************************************************)
CONST  STAR='*';
VAR X,Y,FIRSTNO:INTEGER;
  ASSIGN:STRING[2];
  CH:CHAR;
```

```
(*==============================================================*)
PROCEDURE SELECTTYPE;
(*==============================================================*)
BEGIN
   CASE ASSIGN[1] OF
     '0', '1': BEGIN
                 TITRTYPE:=STRONGACID;
                 SOLSTR:='HYDROCHLORIC ACID';
                 UNKNOWN:=ACID;
               END;
     '2','3': BEGIN
                 TITRTYPE:=STRONGACID;
                 SOLSTR:='SODIUM HYDROXIDE';
                 UNKNOWN:=BASE;
               END;
     '4','5': BEGIN
                 TITRTYPE:=WEAKACID;
                 SOLSTR:='ACETIC ACID';
                 UNKNOWN:=ACID;
               END;
     '6','7': BEGIN
                 TITRTYPE:=WEAKBASE;
                 SOLSTR:=' AMMONIA';
                 UNKNOWN:=BASE;
               END;
     '8','9': BEGIN
                 TITRTYPE:=DIPROTIC;
                 SOLSTR:='OXALIC ACID';
                 UNKNOWN:=ACID
               END;
     END; (*CASE*)
   PAGE(OUTPUT);
END; (* SELECTTYPE *)

BEGIN (* SELECTUNKNOWN *)
 PAGE(OUTPUT);
 X:=0; Y:=6;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X,Y),'ENTER ASSIGNMENT NO.(0-99):'); Y:=Y+2;
 WRITE(AT(X,Y),'This no. will be given to you by'); Y:=Y+1;
 WRITE(AT(X,Y),'your teacher.'); Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,STAR));
 GETRESPONSE(X+35,Y-5,ASSIGN,2,NUMS+['Q']);
 QUIT:=POS('Q',ASSIGN)>0;
 IF QUIT THEN
   BEGIN
     BACKTOMENU;
     EXIT(ASSIGNMENT);
   END;
 IF LENGTH(ASSIGN)<2 THEN ASSIGN:=CONCAT('0',ASSIGN);
 SELECTTYPE;
 FIRSTNO:=(ORD(ASSIGN[1])-48)*3;
 UNKNOWNC:=ORD(ASSIGN[2])-48;
 UNKNOWNC:=(100+(96*UNKNOWNC)+FIRSTNO)/1000;
             (*calculates conc.in rang 0.1 - 0.91*)
 IF UNKNOWN=ACID THEN HCONC:=UNKNOWNC ELSE OHCONC:=UNKNOWNC;
```

```
 PAGE(OUTPUT);
 Y:=4;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X,Y),' YOUR ASSIGNMENT IS TO CALCULATE'); Y:=Y+2;
 WRITE(AT(X,Y),' THE CONCENTRATION OF A SOLUTION'); Y:=Y+2;
 WRITE(AT(X,Y),'   OF ',SOLSTR,'.'); Y:=Y+2;
 WRITE(AT(X,Y),' CONCENTRATION WILL BE IN RANGE');Y:=Y+2;
 WRITE(AT(X,Y),'        0.100-1.000M');Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,STAR));
 WRITE(AT(10,22),'Press <SPACE BAR> to continue ');
 GETACHAR(CH,[SPACE,'Q']);
 QUIT:=CH='Q';
 PAGE(OUTPUT);
END; (* SELECTUNKNOWN *)

(***********************************************************)
PROCEDURE STANDARDSOLN;
(***********************************************************)
CONST  STAR='*';
       X=0;  BLANK='       ';
VAR  S:STRING; CH:CHAR;
     OK:BOOLEAN;
     Y, J,INTNUM : INTEGER;
BEGIN
  IF UNKNOWN=BASE THEN S:='hydrocholoric acid.' ELSE S:='sodium hydroxide.';
  Y:=8;
  WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
  WRITE(AT(X,Y),'Enter concentration of standard'); Y:=Y+2;
  WRITE(AT(X,Y),'solution of ',S);Y:=Y+2;
  WRITE(AT(X,Y),'(0.100-1.000): ');Y:=Y+2;
  WRITE(AT(X,Y),AROW(40,STAR));
  S:='';
  REPEAT
    GETRESPONSE(20,Y-2,S,5,NUMS+['Q']);
    QUIT:=S='Q';
    IF NOT QUIT THEN
      BEGIN
        STDSOLN:=RVALUE(S);
        OK:=((STDSOLN>=0.10) AND (STDSOLN<=1.0));
        IF OK THEN INTNUM:=TRUNC(STDSOLN*100);
        IF NOT OK THEN WRITE(AT(34,Y),BLANK);
      END;
  UNTIL OK OR QUIT;
  IF UNKNOWN=ACID THEN OHCONC:=STDSOLN ELSE HCONC:=STDSOLN;
  PAGE(OUTPUT);
 END; (* STANDARDSOLN *)

(***********************************************************)
PROCEDURE GETCONDITIONS;
(***********************************************************)
VAR ASTR: SHORTSTR;
    PROMPT1,PROMPT2:STRING;
```

```
(*===============================================================*)
PROCEDURE SELECTVOL(VAR FLASKVOL : REAL);
(*===============================================================*)
CONST  MIN=10.0; MAX=50;        (*  range of volume*)
       X=215; Y=10;            (*  coord. to enter input *)
BEGIN (*SELECTVOL*)
  IF INFLASK=ACID THEN PROMPT1 :='acid' ELSE PROMPT1 :='base';
  PROMPT1 :=CONCAT('Select vol. of ',prompt1,' in flask:');
  PROMPT2 :=' (10-50ml)';
  TWOPROMPTS(PROMPT1,PROMPT2);
  INRANGERESPONSE(FLASKVOL,ASTR,MIN,MAX,X,Y);
  CHARTYPE(6);
  TWOPROMPTS(PROMPT1,PROMPT2); (* erase *)
  CHARTYPE(10);
END; (*SELECTVOL*)

(*===============================================================*)
PROCEDURE SELECT(VAR INFLASK: ACIDORBASE);
(*===============================================================*)
VAR CH:CHAR;
BEGIN
 IF ((UNKNOWN=BASE) OR (TITRTYPE=DIPROTIC)) THEN
  BEGIN
   PROMPT1 :='Acidic';
   INFLASK :=ACID;
  END
  ELSE
   BEGIN
    PROMPT1 :='Basic';
    INFLASK :=BASE;
   END;
 PROMPT1 :=CONCAT(PROMPT1,' solution is in flask');
 PROMPT2 :='        Press <SPACE BAR> to continue';
 TWOPROMPTS(PROMPT1,PROMPT2);
 GETACHAR(CH,[SPACE,'Q']);
 QUIT :=CH='Q';
 CHARTYPE(6);
 TWOPROMPS(PROMPT1,PROMPT2);  (* erase*)
 CHARTYPE(10);
END; (*SELECT*)

BEGIN (* GETCONDITIONS *)
 IF UNKNOWN=ACID THEN
  BEGIN
   REALSTR(OHCONC,ASTR,3,5);
   ACIDMOLARITY('?');
   BASEMOLARITY(ASTR);
  END
  ELSE
   BEGIN
    REALSTR(HCONC,ASTR,3,5);
    BASEMOLARITY('?');
    ACIDMOLARITY(ASTR);
   END;
  IF NOT QUIT THEN SELECT(INFLASK);
  IF NOT QUIT THEN SELECTVOL(FLASKVOL);
```

```
END;(* GETCONDITIONS *)

(**************************************************************)
PROCEDURE SHOWTYPE;
(**************************************************************)
VAR X,Y: INTEGER;
      S1 : STRING[18];
BEGIN
   IF UNKNOWN=BASE THEN S1 :='HYDROCHLORIC ACID'  ELSE S1 :='SODIUM HYDROXIDE';
    X:=210; Y:=135;
    WSTAT(X-(7*LENGTH(SOLSTR) DIV 2),Y,SOLSTR);   Y:=Y-20;
    WSTAT(X,Y,'VS');  Y:=Y-20;
    WSTAT(X-(7*LENGTH(S1) DIV 2),Y,S1);
    IF INFLASK=ACID THEN S1 :='BASE' ELSE S1 :='ACID';   Y:=Y-50;
    WSTAT(X-(7*8),Y,CONCAT(S1,' IN BURETTE'));
END; (* SHOWTYPE *)

(*$I :ASS2*)
```

```
(* ASS2.TEXT to be included with ASSIGNMENT *)
(*************************************************************)
PROCEDURE TITRATE;
(*************************************************************)
VAR
   ACIDCOL,BASECOL,                            (* soln colour during titration   *)
   SOLNCOL : SCREENCOLOR;                      (*current colour of soln in flask  *)
   SPACEPR,                                    (* flag to indicate space bar pressed *)
   SELECTCHANGE: BOOLEAN;    (* flag to indicate change in titrant increment volume *)
   INCR,                                       (* current titrant increment vol.  *)
   BURVOL,                                     (* total vol. added from burette.  *)
   ACIDVOL,BASEVOL,                            (* total vol. of acid & base in flask *)
   ENDPT1,                                     (* vol. of titrant required to reach endpt*)
   PH,                                         (* current pH of solution   *)
   XTRAVOL: REAL;            (* vol. of titrant added but not yet shown to fill flask *)
   INTENDPT,                                   (* integer value of endpt*100   *)
   RTSIDE,LTSIDE,                              (* current x-coord. of flask being filled*)
   OLDLEVEL,INCREASE,                          (* current & increase in level of soln  *)
   OLDX,OLDY,                                  (* current coord of pH graph   *)
   INDEX: INTEGER;                             (* required for filling flask   *)

   (*=========================================================*)
   PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
   (*=========================================================*)
   CONST  X=160; Y=150;
   VAR   ASTR: SHORTSTR;
   BEGIN
      IF COLOUR THEN
        BEGIN  ACIDCOL:=VIOLET; BASECOL:=BLUE;  END
          ELSE    (* not colour monitor *)
              BEGIN  ACIDCOL:=WHITE1; BASECOL:=BLACK1; END;
      BURVOL:=0.0;
      IF INFLASK=ACID THEN
         BEGIN
           ACIDVOL:=FLASKVOL;
           BASEVOL:=BURVOL;
           ENDPT1:=ACIDVOL*HCONC/OHCONC;
           SOLNCOL:=ACIDCOL;
         END
         ELSE
           BEGIN
              ACIDVOL:=BURVOL;
              BASEVOL:=FLASKVOL;
              ENDPT1:=BASEVOL*OHCONC/HCONC;
              SOLNCOL:=BASECOL;
           END;
      IF ENDPT1<320.0 THEN INTENDPT:=ROUND(ENDPT1 * 100) ELSE INTENDPT:=32000;
      FILLRATE:=2;      (*determines rate at which flask filled*)
      REALSTR(ACIDVOL,ASTR,2,6);
      ACIDISP(ASTR);     (*display acid vol. *)
      REALSTR(BASEVOL,ASTR,2,6);
      BASEDISP(ASTR);    (*display base vol. *)
      CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
      REALSTR(PH,ASTR,2,5);
      DISPLAYPH(ASTR);
   END; (*INITCONDITIONS*)
```

```
(*=================================================================*)
PROCEDURE INITLEVEL;
(*=================================================================*)
(* initialises coordinates of sides of flask & top of flask and level of soln in flask*)
BEGIN
   FLASKTOP:=FLASKY+(3*FLASKSIZ)DIV 4; (*ycoord of top sloping sides of flask*)
   NECKTOP:=FLASKY+FLASKSIZE;      (*ycoord of very top of flask*)
   LTSIDE:=FLASKX-(FLASKSIZ DIV 2)+2; (*calc. coord of sides of*)
   RTSIDE:=FLASKX+(FLASKSIZ DIV 2)-2; (*flask given midpt of base*)
   OLDLEVEL:=FLASKY+1;            (*base of flask-flasky  *)
   INCREASE:=2*ROUND(FLASKVOL/10);(*depth of soln to be initially placed in flask*)
   XTRAVOL:=0.0;  (*initialize increment in titrant *)
END;(*INITLEVEL*)

(*=================================================================*)
PROCEDURE ADDMORE;
(*=================================================================*)
(*increment vol. of titrant & calculate new pH;display new pH & new vol. of titrant*)
CONST BLANK='      ';
VAR  VOL: INTEGER; PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
   BURVOL:=BURVOL+INCR;
   VOL:=ROUND(BURVOL*100);
   BURVOL:=VOL/100.0;
   REALSTR(BURVOL,VOLSTR,2,6);
   CASE INFLASK OF
     ACID: BEGIN
             BASEVOL:=BURVOL;
             BASEDISP(BLANK);
             BASEDISP(VOLSTR);
           END;
     BASE: BEGIN
             ACIDVOL:=BURVOL;
             ACIDISP(BLANK);
             ACIDISP(VOLSTR);
           END;
     END; (* CASE *)
   CALCPH(ACIDVOL,BASEVOL,HCONC,OHCONC,PH);
   DISPLAYPH(BLANK);          (* ERASE OLD PH *)
   REALSTR(PH,PHSTR,2,5);  (* DISPLAY NEW PH *)
   DISPLAYPH(PHSTR);
 END;  (* ADDMORE *)

(*=================================================================*)
PROCEDURE CHECKINDICATOR;
(*=================================================================*)
VAR INTVOL:INTEGER;
```

```
(*-------------------------------------------------------------------*)
PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
(*-------------------------------------------------------------------*)
(* change colour of soln to newcolor *)
VAR CURRENTL,DEPTH : INTEGER;
BEGIN
   SOLNCOL:=NEWCOLOR;
   CURRENTL:=OLDLEVEL;
   INITLEVEL;
   DEPTH:=CURRENTL-OLDLEVEL;
   FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
   IF CURRENTL>OLDLEVEL THEN
    BEGIN
     DEPTH:=CURRENTL-OLDLEVEL;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
    END;
 END; (*CHANGECOL*)

BEGIN (* CHECKINDICATOR*)
  INTVOL:=ROUND(BURVOL*100);
  IF TITRTYPE=DIPROTIC THEN
    BEGIN
      IF SOLNCOL=ACIDCOL THEN
        BEGIN
          IF (INTVOL DIV 2) >=INTENDPT THEN CHANGECOL(BASECOL);
        END;
    END (*DIPROTIC*)
    ELSE (* NOT DIPROTIC *)
  IF INTVOL>=INTENDPT THEN
    BEGIN
     CASE INFLASK OF
       ACID  : IF (SOLNCOL=ACIDCOL) THEN CHANGECOL(BASECOL);
       BASE  : IF (SOLNCOL=BASECOL) THEN CHANGECOL(ACIDCOL);
       END; (*CASE*)
     END;
END; (* CHECKINDICATOR*)

(*===================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*===================================================================*)
VAR EXTRA : INTEGER;
BEGIN
 XTRAVOL:=XTRAVOL+INCR;(*xtravol is vol.titrant added that has not yet been
                                                         shown to fill flask*)
  IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled by
                         an extra amt. This amt must be even due to slope of flask *)
   BEGIN
    EXTRA:=TRUNC(XTRAVOL/FILLRATE);
    IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
    FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
    XTRAVOL:=0;
   END;
END; (*CHECKLEVEL*)
```

```
BEGIN   (* TITRATE *)
  INITCONDITIONS(ENDPT1);
  INITLEVEL;
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
  AGAIN:=TRUE;
  SELECTINCR(INCR);
  IF NOT QUIT THEN
    BEGIN
      REQUEST;       (*display prompt to press space bar*)
      SELECTCHANGE:=FALSE;
       REPEAT
         CHECKKEY(SPACEPR,SELECTCHANGE);
         IF SPACEPR THEN
           BEGIN
             MOVEDROP(INCR,OLDLEVEL);
             ADDMORE;
             CHECKINDICATOR;
             CHECKLEVEL(XTRAVOL,INCR);
           END;
         IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
      UNTIL QUIT;
     CHARTYPE(0);
     REQUEST;           (*erase prompt*)
     CHARTYPE(10);
  END;
END; (* TITRATE *)

(****************************************************************)
PROCEDURE STARTAGAIN;
(****************************************************************)
CONST BLANK='     ';
       X=0;
VAR   CH:CHAR; Y: INTEGER;
       UNSTR: SHORTSTR;

   (*=============================================================*)
   PROCEDURE CHECKAGAIN;
   (*=============================================================*)
   CONST DOTS='......(';
   BEGIN
    Y:=6;
    WRITE(AT(X,Y),'REPEAT previous titration     ',DOTS,'R)'); Y:=Y+2;
    WRITE(AT(X,Y),'Repeat titration');    Y:=Y+1;
    WRITE(AT(X,Y),'    but ALTER conditions     ',DOTS,'A)'); Y:=Y+2;
    WRITE(AT(X,Y),'Get CALCULATOR               ',DOTS,'C); Y:=Y+2;
    WRITE(AT(X,Y),'New assignment               ',DOTS,'N)'); Y:=Y+2;
    WRITE(AT(X,Y),'QUIT - back to MAIN MENU     ',DOTS,'Q)'); Y:=Y+3;
    WRITE(AT(X,Y),'         SELECT  OPTION   ..',DOTS,'  )');
    GOTOXY(37,Y);
    GETTEXTCHAR(X+37,Y,CH,['R','A','C','N','Q']);
    IF AGAIN THEN AGAIN:=(CH='R') OR (CH='C');
    QUIT:=CH='Q';  (* resets 'quit'  *)
    PAGE(OUTPUT);
    CLEARVALUES(AGAIN);
    FILLBOX(150,271,32,164,BLACK1); (*erase right hand box*)
    CASE CH OF
```

```
        'C' :BEGIN GETCALCULATOR; CHECKAGAIN; END;
        'N':NEW:=TRUE;
        END; (*CASE*)
     PAGE(OUTPUT);
    END; (* CHECKAGAIN*)


BEGIN (* STARTAGAIN *)
   PAGE(OUTPUT);
   NEW:=FALSE;
   CHECKAGAIN;
 END; (* STARTAGAIN *)


BEGIN   (* main *)
  SETCHAIN(':MENU');
  AGAIN:=FALSE;
  NEW:=TRUE;
  QUIT:=FALSE;
  INITSCREEN;
  SETCOLOUR;
  WHILE (NOT QUIT) DO
   BEGIN
       IF NEW THEN SELECTUNKNOWN;
       IF ((NOT AGAIN) AND (NOT QUIT)) THEN STANDARDSOLN;
       DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
       GRAFMODE;
       IF ((NOT QUIT) AND (NOT AGAIN)) THEN
         BEGIN
            GETK(K1,K2);
            IF NOT QUIT THEN GETCONDITIONS;
         END; (* if *)
       SHOWTYPE;
       IF NOT QUIT THEN TITRATE;
       TEXTMODE;
       STARTAGAIN;
  END; (* while *)
  BACKTOMENU;
END. (*ASSIGNMENT*)
```

```
(* This program is SYSTEM.STARTUP for SALTS TITRATION PACKAGE.
 This introduction - displays title page;
                   - asks user whether colour monitor is available.
                   - informs user that input must be followed by return key;
                   - informs user to press "Q"  to quit from any of the programs;
                   - and then chains to main menu.*)
```

The pascal code for the introduction/startup program for the SALTS TITRATION PACKAGE is identical to the code for the introduction program for the ACID/BASE TITRATION PACKAGE with the exception of two lines in Procedure Titlepage:

```
BEGIN (* TITLEPAGE *)
  INITCONDITIONS;
  INITLEVEL;
  DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE2);
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
  X:=150;
  BORDER(VIOLET);
   WSTAT(130,150,'T I T R A T I O N');
   WSTAT(134,130,'O F   S A L T S');
  WSTAT(X,71,'By');
  WSTAT(X,49,'Roslyn Atkins,');
  WSTAT(X,37,'Chemistry Dept.');
  WSTAT(X,25,'Wollongong Uni.');
  WAIT(250);
  GRAFMODE;
  L:=0;
  REPEAT
    L:=L+1;
    CASE L OF
     1,6,11 : NEWCOL:=BLUE;
     2,7,12 : NEWCOL:=WHITE2;
     3,8,13 : NEWCOL:=ORANGE;
     4,9,14 : NEWCOL:=WHITE2;
     5,10,15 :NEWCOL:=VIOLET;
    END; (*CASE*)
    CHANGECOL(NEWCOL);
    WAIT(250);
  UNTIL (L=15) OR (KEYIN);
 END; (* TITLEPAGE*)
```

```
(* Main menu which chains to
   - titration of SALTS (SALTITRATE)
   - titration of SODIUM CARBONATE & SODIUM BICARBONATE(MITURE)
   - assignment of salts  (SALTASSIGN)
   - asssignment of mixture(MIXASSIGN)  *)

(*$S++,R-,V-*)
PROGRAM SALTMENU;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;
VAR
   PROGNUM:CHAR;
   COLOUR:BOOLEAN;
   QUIT: BOOLEAN;

(********************************************************************)
PROCEDURE SELECTOPTION(Y:INTEGER; VAR CH:CHAR; LEGALSET:CHARSET);
(********************************************************************)
BEGIN
    WRITE(AT(0,Y),'         SELECT OPTION ..........(  )');
    GETTEXTCHAR(37,Y,CH,LEGALSET);
    QUIT:=CH='Q';
END;

(********************************************************************)
PROCEDURE SHOWMENU(VAR NUM: CHAR);
(********************************************************************)
CONST  STAR='*';
         DOTS='  .....(';
        TITR='Titration ';
        AORB='of acids & bases';
        BLANK='       ';
        X=0;
VAR   Y : INTEGER;
BEGIN
  PAGE(OUTPUT);
  Y:=0;
  WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
  WRITE(AT(X+10,Y),'M A I N   M E N U'); Y:=Y+2;
  WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+3;
  WRITE(AT(X,Y),TITR,'of salts   ',BLANK,DOTS,'1)'); Y:=Y+2;
  WRITE(AT(X,Y),TITR,'of salt mixture    ',DOTS,'2)'); Y:=Y+2;
  WRITE(AT(X,Y),'Assignment for salts ',BLANK,DOTS,'3)'); Y:=Y+2;
  WRITE(AT(X,Y),'Assignment for mixture      ',DOTS,'4)'); Y:=Y+2;
  WRITE(AT(X,Y),'QUIT        ',BLANK,BLANK,DOTS,'Q)'); Y:=Y+3;
  SELECTOPTION(Y,NUM,['1'..'4','Q']);
 END;  (* SHOWMENU *)
```

```
(*************************************************************)
PROCEDURE CHAINTO(CH: CHAR);
(*************************************************************)
 VAR S:STRING;

    (*=========================================================*)
    PROCEDURE INFORM(NAME:STRING);
    (*=========================================================*)
    BEGIN
        PAGE(OUTPUT);
        WRITE(AT(8,8),'L O A D I N G');
        WRITE(AT(0,12),NAME,'  P R O G R A M .....');
    END;

BEGIN  (* CHAINTO  *)
   CASE CH OF
     '1': BEGIN  S:='T I T R A T I O N';  SETCHAIN(':SALTITRATE'); END;
     '2': BEGIN S:='T I T R A T I O N';   SETCHAIN(':MIXTURE'); END;
     '3': BEGIN S:='A S S I G N M E N T';SETCHAIN(':SALTASSIGN'); END;
     '4': BEGIN S:='A S S I G N M E N T'; SETCHAIN(':MIXASSIGN'); END;
     END;(*CASE*)
   INFORM(S);
  END; (* SELECTPROG *)

(*************************************************************)
PROCEDURE FIN;
(*************************************************************)
VAR X,Y: INTEGER;
BEGIN
   PAGE(OUTPUT);
   X:=5; Y:=6;
   WRITE(AT(X+2,Y),'REMOVE DISK FROM DISK DRIVE'); Y:=Y+5;
   WRITE(AT(X,Y),'IT WILL BE NECESSARY TO REBOOT'); Y:=Y+2;
   WRITE(AT(X,Y),'COMPUTER TO RUN ANOTHER PROGRAM       ');
   REPEAT
      X:=Y; (* INFINITE LOOP*)
   UNTIL (X>Y);
END; (* FIN *)


BEGIN (* MAIN *)
  SETCOLOUR;
  SWAPGPON;  (* set swapping to level 2 *)
  SHOWMENU(PROGNUM);
  IF QUIT THEN FIN  ELSE  CHAINTO(PROGNUM);
END. (*SALTMENU*)
```

```
(* Simulated titration between
  - salt of a weak acid/strong base and a strong acid
  - salt of a weak base/strong acid and a strong base
  - salt of a diprotic acid/strong base and a strong acid  *)
(*$S++*)(*$R-*)(*$V-*)

PROGRAM SALTTITRATE;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,SALTLIB;
CONST
   XCON=160; YCON=40;                          (*coord. of origin of ph graph *)
   VOLSCALE=100;                    (*no. pixels on horizontal axis of ph graph *)
TYPE
   REALPTS=ARRAY[1..VOLSCALE] OF REAL;
   INTPTS=ARRAY[0..VOLSCALE] OF INTEGER;
VAR
   KOPTION,                         (*option for user to enter pK values(s)     *)
   COLOUR,                            (*is colour monitor available             *)
   AGAIN : BOOLEAN;                             (*option to repeat titration *)
   VOLPTS : REALPTS;                (*vol.of titrant used initially to plot curve*)
   PHPTS :  INTPTS;     (*pH values corresponding to volpts required to plot entire curve-
                these integer values have been scaled for graph by a factor of phratio*)
   SALTCONC,TITRCONC :REAL;                       (* conc. of solns               *)

(***********************************************************************)
PROCEDURE GETK(VAR K1,K2 :REAL);
(***********************************************************************)
(* allows user to input pK value(s)  *)
VAR  MIN,MAX : REAL;
     OK:BOOLEAN;

   (*=================================================================*)
   PROCEDURE INPUTK(S:STRING; VAR K:REAL);
   (*=================================================================*)
   VAR PKISTR : SHORTSTR;
        PROMP : STRING;
        PK:REAL;
        X,Y :INTEGER;
    BEGIN (* INPUTK*)
      GRAFMODE;
      X:=125; Y:=10;
      PROMP:=CONCAT('Enter ',S);
      WSTAT(2,10,PROMP);
      INRANGERESPONSE(PK,PKISTR,MIN,MAX,X,Y);
      REMOVERESPONSE(X,Y,LENGTH(PKISTR));
      CHARTYPE(6);
      WSTAT(2,10,PROMP);
      CHARTYPE(10);
      IF NOT QUIT THEN K:=EXP(-PK*LN(10)) ELSE K:=1.0;
    END; (* INPUTK *)
```

```
BEGIN (*GETK*)
   MIN:=2.0; MAX:=15.0;
   CASE TITRTYPE OF
     BASESALT : INPUTK('pKa (2-15)',K1);
     ACIDSALT : INPUTK('pKb (2-15)',K1);
     DISALT : BEGIN
                 REPEAT
                   OK:=FALSE;
                   MAX:=11.0;
                   INPUTK('pK1 (2-11)',K1);
                   MAX:=15.0;
                   IF NOT QUIT THEN INPUTK('pK2 (<15)',K2);
                   IF (NOT QUIT) AND (K1<K2) THEN
                     BEGIN
                       BEEP;BEEP;
                       WSTAT(2,0,'pK2 MUST BE > pK1');
                     END
                     ELSE OK:=TRUE;
                 UNTIL (QUIT) OR (OK);
           CHARTYPE(0);
           WSTAT(2,0,'pK2 MUST BE > pK1');
           CHARTYPE(10);
          END;
    END;(*CASE*)
  KOPTION:=FALSE;
END;(* GETK *)

(************************************************************)
PROCEDURE WHICHSALT(SALTTYPE : TITRAT; VAR FINISHED:BOOLEAN);
(************************************************************)
VAR REPLY:CHAR;
    PROMPT1,PROMPT2,ATYPE,ACH : STRING;
    NUM:STRING[1];

   (*=======================================================*)
   PROCEDURE GETREPLY(VAR REPLY :CHAR; INPUT:CHARSET);
   (*=======================================================*)
   VAR   X,Y: INTEGER;
   BEGIN
     PAGE(OUTPUT);
     X:=0; Y:=2;
     PROMPT1:=CONCAT(PROMPT1,'        .....(1)');
     PROMPT2:=CONCAT(PROMPT2,'        .....(2)');
     WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
     WRITE(AT(2,Y),'Select salt to be used in titration'); Y:=Y+2;
     WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+3;
     WRITE(AT(X,Y),PROMPT1); Y:=Y+2;
     IF TITRTYPE<>ACIDSALT THEN
       BEGIN
         WRITE(AT(X,Y),PROMPT2); Y:=Y+2;
       END;
     WRITE(AT(X,Y),'Input pK',ACH,' of ',ATYPE,' from'); Y:=Y+1;
     PROMPT1:=CONCAT('which salt is derived        .....(',NUM,')');
     WRITE(AT(X,Y),PROMPT1); Y:=Y+3;
     WRITE(AT(X,Y),'      SELECT OPTION      .......(  )');
     GETTEXTCHAR(X+37,Y,REPLY,INPUT);
```

```
    PAGE(OUTPUT);
   END; (*GETREPLY*)

BEGIN (*WHICHSALT*)
 KOPTION:=FALSE;
 CASE SALTTYPE OF
   BASESALT: BEGIN
                 PROMPT1:='Sodium cyanide      ';
                 PROMPT2:='Sodium acetate      ';
                 ACH:='a'; ATYPE:='weak acid'; NUM:='3';
                 GETREPLY(REPLY,['1'..'3','Q']);
                 CASE REPLY OF
                  '1': K1:=5.00E-10;
                  '2': K1:=1.76E-5;
                  '3': KOPTION:=TRUE;
                  END; (*CASE*)
                END;
    ACIDSALT: BEGIN
                 PROMPT1:='Ammonium chloride   ';
                 PROMPT2:=' ';
                 ACH:='b'; ATYPE:='weak base'; NUM:='2';
                 GETREPLY(REPLY,['1'..'2','Q']);
                 IF REPLY='1' THEN K1:=1.79E-5
                  ELSE IF REPLY='2' THEN KOPTION:=TRUE;
                END;
      DISALT: BEGIN
                 PROMPT1:='Sodium carbonate    ';
                 PROMPT2:='Potassium phthalate ';
                 ACH:='1 & 2';  ATYPE:='acid'; NUM:='3';
                 GETREPLY(REPLY,['1'..'3','Q']);
                 CASE REPLY OF
                   '1': BEGIN  K1:=4.3E-7; K2:=5.6E-11;  END;
                   '2': BEGIN  K1:=1.3E-3; K2:=3.9E-6;   END;
                   '3': KOPTION:=TRUE;
                  END;(*CASE*)
                END;
     END; (* CASE *)
  FINISHED:=REPLY<>'Q';
END; (*WHICHSALT*)
```

```
(*************************************************************)
PROCEDURE SELECTTYPE(VAR TITRTYPE : TITRAT);
(*************************************************************)
(* select type of titration - then select strength of salt in WHICHSALT *)
VAR OK:BOOLEAN;

   (*==========================================================*)
   PROCEDURE GETTYPE;
   (*==========================================================*)
   CONST   STAR='*';  DOTS=' ..(';
   VAR CH:CHAR;
        X, Y:INTEGER;      (* coord. to enter input *)
   BEGIN
     PAGE(OUTPUT);
     X:=0; Y:=2;
     WRITE(AT(X,Y),AROW(40,STAR));  Y:=Y+2;
     WRITE(AT(X+9,Y),'TITRATION OF SALTS ');  Y:=Y+2;
     WRITE(AT(X,Y),AROW(40,STAR));  Y:=Y+3;
     WRITE(AT(X,Y),'Salt of weak acid /strong base   ',DOTS,'1)');  Y:=Y+2;
     WRITE(AT(X,Y),'Salt of weak base /strong acid   ',DOTS,'2)');  Y:=Y+2;
     WRITE(AT(X,Y),'Salt of diprotic acid/strong base',DOTS,'3)');  Y:=Y+2;
     WRITE(AT(X,Y),'Quit - back to MAIN MENU         ',DOTS,'Q)');  Y:=Y+4;
     WRITE(AT(X,Y),'        SELECT SALT       .......(  )');
     GETTEXTCHAR(X+37,Y,CH,['1'..'3','Q']);
     QUIT:=CH='Q';
     PAGE(OUTPUT);
     CASE CH OF
       '1': TITRTYPE:=BASESALT;
       '2': TITRTYPE:=ACIDSALT;
       '3': TITRTYPE:=DISALT;
      END; (*CASE*)
     END; (*GETTYPE*)

BEGIN (* SELECTTYPE*)
   REPEAT
     GETTYPE;
     IF NOT QUIT THEN WHICHSALT(TITRTYPE,OK);
   UNTIL OK OR QUIT;
END; (* SELECTTYPE *)

(*$I :SALTITR2*)
(*$I :SALTITR3*)
```

```
(* SALTITR2 - included in SALTITRATE*)
(**************************************************************)
PROCEDURE TITRATE;
(**************************************************************)
VAR
   SALTCOL,TITRCOL,MIDCOL,                         (* soln colours during titration  *)
   SOLNCOL : SCREENCOLOR;                          (* current colour of soln in flask  *)
   FIRSTDR,                        (*flag set at beginning of titration and at endpt(s) to
                                   indicate that next drop will require a change in label *)
   LABELS,                                  (* option to show "state" of soln in flask*)
   SPACEPR,                                  (* flag to indicate space bar pressed *)
   SELECTCHANGE : boolean;      (* flag to indicate change in titrant increment volume *)
   INCR,                                      (* current titrant increment vol.  *)
   BURVOL,                              (* total vol. added from burette (titrant)  *)
   SALTVOL,TITRVOL,                        (* total vol. of salt & titrant in flask  *)
   ENDPT1,                              (* vol. of titrant required to reach endpt  *)
   PH,                                                (* current pH of soln  *)
   NEXTVOL,                                  (*nextvol required for graphing  *)
   XTRAVOL,                             (* vol. titrant not yet shown to fill flask *)
   PHRATIO : REAL;        (*ratio between pHscale & pH range to be plotted on scale*)
   INTENDPT,       (* integer value of endpt*100 required for indicator colour change*)
   RTSIDE,LTSIDE,                          (* current x-coord. of flask being filled  *)
   OLDLEVEL,INCREASE,                       (* current & increase in level of soln *)
   OLDX,OLDY,                                       (* current coord of pH graph  *)
   INDEX :INTEGER;                                (* required for graphing pH curve *)
   NEWEQ : SHORTSTR;                              (* label of "state" of soln in flask *)

    (*============================================================*)
    PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
    (*============================================================*)
    CONST PHSCALE=100.0;  (* No. pixels on pH scale*)
          PHRANGE=14.0;   (* pH 0 - 14 equally spaced on pHscale*)
    VAR  VOLSTR: SHORTSTR;
         X,Y :INTEGER;
    BEGIN
     X:=XCON+8;
     Y:=YCON+110;
     IF COLOUR THEN
       BEGIN  SALTCOL:=VIOLET; TITRCOL:=BLUE; MIDCOL:=GREEN;  END
       ELSE (* not colour monitor *)
        BEGIN
         SALTCOL:=WHITE1;
         IF TITRTYPE=DISALT THEN
          BEGIN  MIDCOL:=BLACK1;  TITRCOL:=ORANGE;   END
          ELSE TITRCOL:=BLACK1;
        END;
     BURVOL:=0.0;
     IF INFLASK=ASALT THEN
        BEGIN
          SALTVOL:=FLASKVOL;
          TITRVOL:=BURVOL;
          ENDPT1:=SALTVOL*SALTCONC/TITRCONC;
          SOLNCOL:=SALTCOL;
          IF TITRTYPE=ACIDSALT THEN WSTAT(X,Y,'pH vs vol.base')
            ELSE WSTAT(X,Y,'pH vs vol.acid');
        END
```

```
  ELSE (*acid or base in flask*)
   BEGIN
    SALTVOL:=BURVOL;
    TITRVOL:=FLASKVOL;
    ENDPT1:=TITRVOL*TITRCONC/SALTCONC;
    SOLNCOL:=TITRCOL;
    WSTAT(X,Y,'pH vs vol.salt');
   END;

  IF ENDPT1<320.0 THEN INTENDPT:=ROUND(ENDPT1 * 100)
   ELSE INTENDPT:=32000;(* intendpt required to change indicator- Due to vol.
                        of flask,if endpt>320 then titration will not reach endpt*)
  PHRATIO:=(PHSCALE/PHRANGE); (* pH increm. per pixel *)
  FILLRATE:=2;  (*determines rate at which flask filled*)
  REALSTR(SALTVOL,VOLSTR,2,6);
  SALTDISP(VOLSTR);          (* Display acid volume*)
  REALSTR(TITRVOL,VOLSTR,2,6);
  TITRDISP(VOLSTR);          (* Display base volume *)
 END;  (*INITCONDITIONS*)

(*===================================================================*)
PROCEDURE INITGRAPH;
(*===================================================================*)
BEGIN
 OLDX:=XCON;OLDY:=PHPTS[0]+YCON;
 INDEX:=0; NEXTVOL:=VOLPTS[1];
END;

(*===================================================================*)
PROCEDURE INITLEVEL;
(*===================================================================*)
(* initializes coord of sides of flask & top of flask as well as level of soln in flask *)
CONST WIDTH=2; (* indent soln from sides of flask *)
BEGIN
 FLASKTOP:=FLASKY+ (3*FLASKSIZ)DIV 4; (*y-coord. of top sloping sides of flask*)
 NECKTOP:=FLASKY+FLASKSIZE;   (*y-coord. of very top of flask *)
 LTSIDE:=FLASKX-(FLASKSIZ DIV 2)+WIDTH; (*calc. coord of sides*)
 RTSIDE:=FLASKX+(FLASKSIZ DIV 2)-WIDTH; (*of flask given midpt. of base*)
 OLDLEVEL:=FLASKY+1;             (* base of flask= Flasky  *)
 INCREASE:=10;  (*depth of soln to be initially placed in flask*)
 XTRAVOL:=0.0;          (*initialize increment in titrant *)
END; (*INITLEVEL*)

(*===================================================================*)
PROCEDURE UPDATEQ;
(*===================================================================*)
(* Update current string relating to titration*)
CONST  WACID=' W.ACID'; WBASE=' W.BASE'; SBASE=' BASE ';SSALT=' ASALT ';
       SACID=' ACID  '; ABUFFER=' BUFFER';EQUIV=' END PT';
       (* strings to be displayed at appropriate stages of titration*)
       X=40; Y=27;  (* coord. at which string displayed*)
VAR  INTVOL: INTEGER;
BEGIN
  INTVOL:=ROUND(BURVOL*100);
  FIRSTDR:=FALSE;
```

```
    IF BURVOL=0 THEN
     BEGIN
       IF INFLASK=ASALT THEN NEWEQ:=SSALT
        ELSE
         BEGIN
           CASE TITRTYPE OF
             DISALT, BASESALT:  NEWEQ:=SACID;
             ACIDSALT:          NEWEQ:=SBASE;
           END; (* CASE *)
         END;
      FIRSTDR:=TRUE;
     END
     ELSE (*burvol <>0*)
     BEGIN
      IF TITRTYPE=DISALT THEN
        BEGIN
         IF INTVOL=INTENDPT THEN NEWEQ:=' ENDPT1'
         ELSE  IF (INTVOL DIV 2)=INTENDPT THEN NEWEQ:=' ENDPT2'
         ELSE  IF INTVOL<INTENDPT THEN NEWEQ:=' 2SALTS'
         ELSE  IF (INTVOL DIV 2)<INTENDPT THEN NEWEQ:=ABUFFER
         ELSE  NEWEQ:=SACID;
         FIRSTDR:=((NEWEQ=' ENDPT1') OR (NEWEQ=' ENDPT2'));
        END

       ELSE  (* not diprotic *)
        BEGIN
         IF INTVOL=INTENDPT THEN
          BEGIN  NEWEQ:=EQUIV;  FIRSTDR:=TRUE;  END
          ELSE IF INTVOL<INTENDPT THEN
            BEGIN
             IF INFLASK=ASALT THEN NEWEQ:=ABUFFER;
            END
          ELSE  (* Burvol>endpt*)
             BEGIN
              IF INFLASK=ASALT THEN
               BEGIN
                IF TITRTYPE=ACIDSALT THEN NEWEQ:=SBASE  ELSE NEWEQ:=SACID;
               END
                ELSE NEWEQ:=ABUFFER;
             END; (*ELSE*)
       END;
    END;
   CHARTYPE(0); WSTAT(X,Y,NEWEQ); CHARTYPE(10);
   WSTAT(X,Y,NEWEQ);
END;  (*UPDATEQ*)

(*==================================================================*)
PROCEDURE SETUPARRAYS(VAR VOLPTS:REALPTS;  VAR PHPTS: INTPTS);
(*==================================================================*)
(* calculate pH value for volscale no. points.  Volume calculated is twice required to
reach end point if  monoprotic  and three times if diprotic *)
CONST  MIN=0.01; MAX=10.00;   (*min & max value of incr. of titrant *)
       X=195; Y=10;      (*coord . for display of increment selected*)
VAR  I:INTEGER;
     VOLRATIO:REAL;  (*ratio of vol. of titrant plotted to no.  pixels onx- axis*)
     INCSTR:STRING;
```

```
 CH:CHAR;
 PHSTR:SHORTSTR;

(*-------------------------------------------------------------------------*)
PROCEDURE INITARRAYS;
(*-------------------------------------------------------------------------*)
 BEGIN
    PHPTS[0]:=ROUND(PH*PHRATIO);
    IF TITRTYPE=DISALT   THEN  VOLRATIO:=(ENDPT1*3.0)/VOLSCALE
      ELSE  VOLRATIO:=(ENDPT1*2.0)/VOLSCALE;   (* vol.incr. for each pixel*)
    FOR I:=1 TO VOLSCALE DO VOLPTS[I]:=VOLRATIO*I; (*total vol.at point 'I'*);
END; (* INITARRAYS *)

(*-------------------------------------------------------------------------*)
PROCEDURE INFORMPH;
(*-------------------------------------------------------------------------*)
BEGIN
   WSTAT(3,10,CONCAT('Initial pH is ',PHSTR));
   WSTAT(65,0,'Press <SPACE BAR> to continue');
END; (*INFORMPH*)

(*-------------------------------------------------------------------------*)
PROCEDURE LABELOPTION;
(*-------------------------------------------------------------------------*)
BEGIN
    WSTAT(3,10,'Do you want solution');
    WSTAT(3,0,'in flask labelled?  (Y/N)');
END; (* LABELOPTION *)

(*-------------------------------------------------------------------------*)
PROCEDURE PLEASEWAIT;
(*-------------------------------------------------------------------------*)
BEGIN
    WSTAT(10,5,'PREPARING SOLUTIONS . . . . . .');
END; (*PLEASEWAIT*)

(*-------------------------------------------------------------------------*)
PROCEDURE CYCLE;
(*-------------------------------------------------------------------------*)
BEGIN
    IF NOT AGAIN THEN WHILE ((I<VOLSCALE) AND (NOT KEYIN)) DO
       BEGIN
          I:=I+1;
          IF INFLASK=ASALT THEN
           CALCPH(FLASKVOL,VOLPTS[I],SALTCONC,TITRCONC,PH)
             ELSE  CALCPH(VOLPTS[I],FLASKVOL,SALTCONC,TITRCONC,PH);
          PHPTS[I]:=ROUND(PHRATIO*PH);
        END;(* WHILE *)
 END; (* CYCLE *)

BEGIN (*SETUPARRAYS*)
   CALCPH(SALTVOL,TITRVOL,SALTCONC,TITRCONC,PH);
   REALSTR(PH,PHSTR,2,5);
   IF (NOT AGAIN) THEN INITARRAYS;
    I:=0;
    LABELOPTION;
```

```
CYCLE;
GETHICHAR(X,Y-10,CH,['Y','N','Q']);
REMOVERESPONSE(X,Y-10,1);
LABELS:=CH='Y'; QUIT:=CH='Q';
CHARTYPE(6);LABELOPTION; CHARTYPE(10);
IF LABELS THEN UPDATEQ;
IF QUIT THEN EXIT(SETUPARRAYS);
INCRPROMPT;
CYCLE;
INRANGERESPONSE(INCR,INCSTR,MIN,MAX,X,Y); (*get increment*)
CHARTYPE(6);INCRPROMPT;CHARTYPE(10);(*erase prompt for incr.*)
IF QUIT THEN EXIT(SETUPARRAYS);
INFORMPH;         (*Inform initial pH*)
CYCLE;
GETACHAR(CH,[SPACE,'Q']);
CHARTYPE(6);INFORMPH;CHARTYPE(10);
QUIT:=CH='Q';
IF QUIT THEN EXIT(SETUPARRAYS);

IF ((I<VOLSCALE) AND (NOT AGAIN))THEN
 BEGIN
   PLEASEWAIT;  (*Display prompt*)
   REPEAT
      CYCLE;
      IF KEYIN THEN READ(CH);
   UNTIL I=VOLSCALE;
   CHARTYPE(6);PLEASEWAIT;CHARTYPE(10); (*erase prompt*)
 END;
DISPLAYPH(PHSTR); (* display initial pH*)
END;(* SETUPARRAYS *)
```

```
(* SALTITR3 - included in SALTITRATE*)
   (*===============================================================*)
   PROCEDURE ADDMORE;
   (*===============================================================*)
   (* Increment volume of titrant & calculate new pH *)
   (* Display new pH & new volume of titrant*)
   CONST BLANK='     ';
   VAR VOL : INTEGER;
       PHSTR,VOLSTR:SHORTSTR;
   BEGIN  (* ADDMORE*)
      BURVOL:=BURVOL+INCR;  (* calculate total vol. of titrant   *)
      VOL:=ROUND(BURVOL*100); (* this prevents build up of floating *)
      BURVOL:=VOL/100.0;      (* point errors *)
      REALSTR(BURVOL,VOLSTR,2,6);    (* convert vol.  to string   *)

      CASE INFLASK OF        (* display vol of titrant on screen *)
       ACID, BASE : BEGIN          (* as either vol. of base or acid   *)
                      SALTVOL:=BURVOL;
                      SALTDISP(BLANK);
                      SALTDISP(VOLSTR);
                    END;
        ASALT:      BEGIN
                      TITRVOL:=BURVOL;
                      TITRDISP(BLANK);
                      TITRDISP(VOLSTR);
                    END
        END; (* CASE *)
      CALCPH(SALTVOL,TITRVOL,SALTCONC,TITRCONC,PH);
      DISPLAYPH(BLANK);                (* Erase old pH     *)
      REALSTR(PH,PHSTR,2,5);             (* convert to string *)
      DISPLAYPH(PHSTR);                (* Display new pH    *)
    END;  (* ADDMORE *)

   (*===============================================================*)
   PROCEDURE CHECKINDICATOR;
   (*===============================================================*)
   (* check if volume of titrant added has reached or exceeded end  point and if indicator
   has not yet changed colour then do so *)
   VAR  INTVOL:INTEGER;

      (*-----------------------------------------------------------*)
      PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
      (*-----------------------------------------------------------*)
      (*Change colour of soln to newcolor & change label of soln in flask*)
      VAR CURRENTL,DEPTH : INTEGER;
      BEGIN
           SOLNCOL:=NEWCOLOR;
           CURRENTL:=OLDLEVEL;
           INITLEVEL;
           DEPTH:=CURRENTL-OLDLEVEL;
           FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
           IF CURRENTL>OLDLEVEL THEN  (* colour change with soln in neck of flask*)
             BEGIN
               DEPTH:=CURRENTL-OLDLEVEL;
               FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
             END;
```

```
        IF LABELS THEN UPDATEQ;
     END; (*CHANGECOL*)
BEGIN (*CHECKINDICATOR*)
   INTVOL :=ROUND(BURVOL*100);
   IF TITRTYPE=DISALT THEN
    BEGIN
      IF SOLNCOL=SALTCOL THEN
       BEGIN
        IF (INTVOL DIV 2)>=INTENDPT THEN CHANGECOL(TITRCOL)
        ELSE IF INTVOL>=INTENDPT THEN CHANGECOL(MIDCOL);
       END
       ELSE IF (SOLNCOL=MIDCOL) AND ((INTVOL DIV 2)>=INTENDPT)
          THEN CHANGECOL(TITRCOL);
    END (*DISALT*)

    ELSE (* NOT DIPROTIC *)
      IF (INTVOL>=INTENDPT) THEN
       BEGIN
        CASE INFLASK OF
          ACID, BASE  :IF SOLNCOL=TITRCOL THEN CHANGECOL(SALTCOL);
          ASALT :       IF SOLNCOL=SALTCOL THEN CHANGECOL(TITRCOL);
        END; (*CASE*)
       END;
  END; (* CHECKINDICATOR*)

(*=================================================================*)
PROCEDURE GRAPH(VAR OLDX,OLDY,INDEX : INTEGER; VAR NEXTVOL : REAL;
                                              COL : SCREENCOLOR);
(*=================================================================*)
(*To plot graph use values of pH already scaled & stored as  integers in PHLTS array.
  Plot all points with  titrant volume of less than  or equal to burette volume *)
VAR  EXACTPH,      (*scaled integer pH calc.from current pH value*)
     X,Y : INTEGER;  (*new x,y coord of graph *)
BEGIN
   MOVECOL(OLDX,OLDY,COL);
   WHILE (BURVOL>=NEXTVOL) AND (INDEX<VOLSCALE) DO
    BEGIN
       INDEX :=INDEX+1;
       X :=INDEX+XCON; Y :=PHPTS[INDEX]+YCON;
       MOVETO(X,Y);
       NEXTVOL :=VOLPTS[INDEX+1];
       OLDX :=X;
       OLDY :=Y;
    END;
  IF (INDEX<VOLSCALE) THEN
   BEGIN
      EXACTPH :=ROUND(PH*PHRATIO);
      MOVETO(OLDX,EXACTPH+YCON);
      OLDY :=EXACTPH+YCON;
   END;
  PENCOLOR(NONE);
END; (*GRAPH*)
```

```
(*==================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*==================================================================*)
(*Volume of solution in flask is only shown to increase when a suitable volume (say
5mL or more) has been released from burette.  Therefore smaller increments are
summed until this volume is  reached and then level of soln is shown to rise *)
VAR   EXTRA : INTEGER;
BEGIN
 XTRAVOL :=XTRAVOL+INCR; (*xtravol. is vol.titrant added that has not yet been
                                                          shown  to fill flask*)
 IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled by
                          an extra amt.  This value must be even due to slope of flask*)
   BEGIN
     EXTRA :=TRUNC(XTRAVOL/FILLRATE);
     IF ODD(EXTRA) THEN EXTRA :=EXTRA-1;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
     XTRAVOL :=0;
   END;
END; (*CHECKLEVEL*)


BEGIN    (* TITRATE *)
 INITCONDITIONS(ENDPT1);
 INITLEVEL;
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
 SETUPARRAYS(VOLPTS,PHPTS);
 INITGRAPH;
 IF NOT QUIT THEN
    BEGIN
       REQUEST;         (*Display prompt to press space bar*)
       SELECTCHANGE :=FALSE;
       FIRSTDR :=TRUE;
        REPEAT
          CHECKKEY(SPACEPR,SELECTCHANGE);
          IF SPACEPR THEN
              BEGIN
                 MOVEDROP(INCR,OLDLEVEL);
                 ADDMORE;
                 CHECKINDICATOR;
                 IF ((FIRSTDR) AND (LABELS)) THEN UPDATEQ;
                 CHECKLEVEL(XTRAVOL,INCR);
                 GRAPH(OLDX,OLDY,INDEX,NEXTVOL,WHITE1);
              END;
          IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
        UNTIL QUIT;
       CHARTYPE(0);
       REQUEST;            (*erase prompt*)
       CHARTYPE(10);
       AGAIN :=(BURVOL>0); (* only give option to repeat 'again' if
                        titration has commenced*)
    END;
END; (* TITRATE *)
```

```
(**************************************************************)
PROCEDURE STARTAGAIN;
(**************************************************************)
   (*=========================================================*)
   PROCEDURE NEWLABELS(COL : SCREENCOLOR);
   (*=========================================================*)
   CONST   LOWERY=170; TOPY=190;  (* y coord of small boxes *)

      (*------------------------------------------------------*)
      PROCEDURE LABELS(X: INTEGER; CH1,CH2:SHORTSTR);
      (*------------------------------------------------------*)
      CONST  HEIGHT=179; (*y coord of title of small boxes *)
      BEGIN
         WSTAT(X,HEIGHT,CH1);
         WSTAT(X+7,HEIGHT-7,CH2);
      END;

   BEGIN (*NEWLABELS*)
    FILLBOX(XMIN,XMAX,LOWERY-2,YMAX,BLACK1);
    DRAWBOX(1,LOWERY,64,TOPY,COL);     (* salt molarity *)
    LABELS(5,'M','s');
    DRAWBOX(70,LOWERY,135,TOPY,COL);   (* titr molarity *)
    IF TITRTYPE=ACIDSALT THEN LABELS(76,'M','b') ELSE LABELS(76,'M','a');
    DRAWBOX(145,LOWERY,206,TOPY,COL);  (* salt volume   *)
    LABELS(149,'V','s');
    DRAWBOX(212,LOWERY,275,TOPY,COL);  (* titr volume   *)
    IF TITRTYPE=ACIDSALT THEN LABELS(218,'V','b') ELSE LABELS(218,'V','a');
   END; (* NEWLABELS*)

   (*=========================================================*)
    PROCEDURE CLEARVALUES;    (* erase all values and flask *)
   (*=========================================================*)
   CONST  BLANK='     ';
   BEGIN
     DISPLAYPH(BLANK);        (* erase pH *)
     TITRDISP(BLANK);        (* erase vol. base *)
     SALTDISP(BLANK);         (* erase vol. acid *)
     IF NOT AGAIN THEN
       BEGIN
          NEWLABELS(BLUE);
       END;
      FILLBOX(10,110,25,125,BLACK1);     (* erase flask  *)
  END; (*CLEARVALUES*)

   (*=========================================================*)
   PROCEDURE CHECKAGAIN;
   (*=========================================================*)
   CONST   X=0;       (* coord. to enter input *)
   VAR CH:CHAR; Y:INTEGER;

      (*------------------------------------------------------*)
      PROCEDURE KEEPCURVE;
      (*------------------------------------------------------*)
      CONST  X=0; Y= 10;       (* coord. to enter input *)
      VAR  REPLY:CHAR;
      BEGIN
```

```
        PAGE(OUTPUT);
        WRITE(AT(X,Y),'Do you wish to retain pH curve from');
        WRITE(AT(0,Y+2),'previous titration? (Y/N)');
        GETTEXTCHAR(X+28,Y+2,REPLY,['Y','N','Q']);
        IF REPLY='N' THEN FILLBOX(150,265,32,164,BLACK1);
        QUIT :=(CH='Q');               (* erase pH graph *)
    END; (*KEEPCURVE*)

  BEGIN (* CHECKAGAIN *)
    PAGE(OUTPUT); Y :=6;
    WRITE(AT(X,Y),'Repeat previous titration       .....(R)'); Y :=Y+1 ;
    WRITE(AT(X,Y),'Select different titration      .....(S)');Y :=Y+2;
    WRITE(AT(X,Y),'Quit - back to MAIN MENU        .....(Q)');Y :=Y+3;
    WRITE(AT(X,Y),'          SELECT  OPTION    ........(  )');
    GETTEXTCH(X+37,Y+7,CH,['R','S','Q']);
    AGAIN :=CH='R';
    QUIT :=CH='Q';   (* resets 'quit'  *)
    IF ((NOT QUIT) AND (NOT AGAIN)) THEN
      BEGIN
        KEEPCURVE;
        IF NOT QUIT THEN SELECTTYPE(TITRTYPE);
      END
      ELSE FILLBOX(150,265,32,164,BLACK1); (*erase pH graph*)
    END; (* CHECKAGAIN*)

BEGIN (* STARTAGAIN*)
    IF AGAIN THEN CHECKAGAIN ELSE SELECTTYPE(TITRTYPE);
    CLEARVALUES;
    PAGE(OUTPUT);
 END; (* STARTAGAIN *)


BEGIN   (* main *)
  SETCHAIN(':MENU');
  AGAIN :=FALSE;
  QUIT :=FALSE;
  INITSCREEN;
  SETCOLOUR;
  SELECTTYPE(TITRTYPE);
  WHILE (NOT QUIT) DO
   BEGIN
    DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
    DRAWAXES(XCON,YCON,VOLSCALE,WHITE1);
    GRAFMODE;
    IF (NOT AGAIN) THEN
      BEGIN
        IF KOPTION THEN GETK(K1,K2);
        IF NOT QUIT THEN SETUPCONDITIONS(SALTCONC,TITRCONC,INFLASK);
      END; (* if *)
    IF NOT QUIT THEN TITRATE;
    TEXTMODE;
    STARTAGAIN;
  END; (* while *)
  BACKTOMENU;
END. (*SALTTITRATE*)
```

```
(* Simulated titration between a mixture of sodium carbonate and sodium bicarbonate with
hydrochloric acid *)
(*$S++*)(*$R-*)(*$V-*)

PROGRAM SALTMIXTURE;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,SALTLIB;
CONST
   XCON=160; YCON=40;          (*coord. of origin of ph graph *)
   VOLSCALE=100;   (*no. pixels on horizontal axis of ph graph *)
TYPE
   REALPTS=ARRAY[1..VOLSCALE] OF REAL;
   INTPTS=ARRAY[0..VOLSCALE] OF INTEGER;
VAR
  COLOUR,                                              (*is colour monitor available *)
  AGAIN : BOOLEAN;                                     (*option to repeat titration *)
  VOLPTS: REALPTS;                        (*vol.of titrant used initially to plot curve*)
  PHPTS: INTPTS;       (*pH values corresponding to volpts required to plot entire curve-
               these integer values have  been scaled for graph by a factor of phratio*)
  SALT1CONC,SALT2CONC, TITRCONC: REAL;                               (* conc. of solns *)
  HYPO,                                     (* flag to indicate hypothetical indicator *)
  NEWIND:BOOLEAN;  INDNUM:CHAR;

(**********************************************************************)
PROCEDURE ACIDDISP(S:SHORTSTR);
(**********************************************************************)
BEGIN
   WSTAT(7,148,S); (*display acidconc next to flask*)
   WSTAT(20,140,'HCl');
END; (*ACIDDISP*)

(**********************************************************************)
PROCEDURE ERASEBOXES;
(**********************************************************************)
BEGIN
  FILLBOX(5,40,172,188,BLACK2);
  FILLBOX(76,100,172,188,BLACK2);
END;

(**********************************************************************)
PROCEDURE ERASELABELS;
(**********************************************************************)
BEGIN
  FILLBOX(12,50,172,186,BLACK1);
  FILLBOX(83,125,172,186,BLACK1);
END;

(**********************************************************************)
PROCEDURE NEWLABELS(ANUM:INTEGER);
(**********************************************************************)
VAR X1,X2:INTEGER;
    (*==============================================================*)
    PROCEDURE NEWSTR(X:INTEGER;CH1,CH2:SHORTSTR);
    (*==============================================================*)
    CONST HEIGHT=178;
    BEGIN
       WSTAT(X,HEIGHT,CH1);
```

```
      WSTAT(X,HEIGHT-4,CH2);
    END; (*NEWSTR*)
BEGIN (*NEWLABELS*)
  X1:=15; X2:=85;
  CHARTYPE(6);
  IF ANUM=1 THEN NEWSTR(X1,'Na CO',' 2 3') ELSE NEWSTR(X2,'NaHCO','    3');
  CHARTYPE(10);
END; (*NEWLABELS*)

(****************************************************************)
PROCEDURE SELECTTYPE(VAR TITRTYPE: TITRAT);
(****************************************************************)
(* select type of titration - then select strength of salt in WHICHSALT *)
CONST  STAR='*';  DOTS=' ..(';
VAR CH:CHAR;
    X, Y:INTEGER;      (* coord. to enter input *)
BEGIN
 PAGE(OUTPUT);
 X:=0; Y:=0;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X+9,Y),'MIXTURE OF SALTS '); Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+3;
 WRITE(AT(6,Y),'The solution to be titrated'); Y:=Y+2;
 WRITE(AT(5,Y),'will contain sodium carbonate '); Y:=Y+2;
 WRITE(AT(7,Y),'and sodium bicarbonate.'); Y:=Y+4;
 WRITE(AT(5,Y),'This mixture is titrated with'); Y:=Y+2;
 WRITE(AT(10,Y),'hydrochloric acid.');
 WRITE(AT(10,23),'Press <SPACE BAR> to continue.');
 GETACHAR(CH,[SPACE,'Q']);
 QUIT:=CH='Q';
 TITRTYPE:=DISALT;
 K1:=4.3E-7; K2:=5.6E-11;
END; (* SELECTTYPE *)

(****************************************************************)
PROCEDURE SETCONDITIONS(VAR S1CONC,S2CONC,TCONC:REAL;VAR INFLASKACIDORBASE);
(****************************************************************)
VAR CONCSTR : STRING;
    SCONC:REAL;
    SELECT:INTEGER;
    ZERO:BOOLEAN;
```

```
(*=================================================================*)
PROCEDURE SELECTCONC(VAR CONC : REAL; VAR ACIDBASE :STRING);
(*=================================================================*)
CONST  MAX=1.0;     (* Range of soln concentration *)
       X=160; Y=0;          (* coord. to enter input*)
VAR  PROMPT,(*string for molarity of acid or base*)
     MINSTR:STRING;
     CONCSTR:STRING[6];
     MIN:REAL;
BEGIN       (* SELECTCONC *)
 IF ACIDBASE='hydrochloric acid' THEN
   BEGIN MIN:=0.001; MINSTR:='0.001'; END
     ELSE BEGIN MIN:=0.0; MINSTR:='0.000'; END;
 PROMPT:='Enter conc. of ';
 PROMPT:=CONCAT(PROMPT,ACIDBASE);
 WSTAT(3,Y+10,PROMPT);          (* DISPLAY PROMPT  *)
 MINSTR:=CONCAT('(',MINSTR,' - 1.000M): ');
 WSTAT(3,0,MINSTR);
 INRANGERESPONSE(CONC,ACIDBASE,MIN,MAX,X,Y);
 CHARTYPE(6);
 WSTAT(3,Y+10,PROMPT);          (* erase prompt*)
 WSTAT(3,0,MINSTR);
 CHARTYPE(10);
 REALSTR(CONC,CONCSTR,3,5);
END;        (* SELECTCONC *)

(*=================================================================*)
PROCEDURE SELECTVOL(VAR FLASKVOL : REAL);
(*=================================================================*)
 CONST  MIN=10.0; MAX=50;       (*  range of volume*)
        X=215; Y=10;          (*  coord. to enter input *)
 VAR  PROMPT1,PROMPT2: STRING;
      FLASKSTR: SHORTSTR;
 BEGIN
    PROMPT1:='Select vol. of salt solution:';
    PROMPT2:=' (10.0-50.0mL)';
    TWOPROMPTS(PROMPT1,PROMPT2);
    INRANGERESPONSE(FLASKVOL,FLASKSTR,MIN,MAX,X,Y);
    CHARTYPE(6);
    TWOPROMPTS(PROMPT1,PROMPT2); (* ERASE *)
    CHARTYPE(10);
 END; (*SELECTVOL*)

 (*================================================================*)
 PROCEDURE INFORM;
 (*================================================================*)
 VAR  PROMPT1,PROMPT2:STRING; CH:CHAR;
 BEGIN
   PROMPT1:='The concentration of BOTH salts';
   PROMPT2:='cannot be zero!!  Press <SPACE BAR>';
   TWOPROMPTS(PROMPT1,PROMPT2);
   GETACHAR(CH,[SPACE,'Q']);
   QUIT:=CH='Q';
   CHARTYPE(6);
   TWOPROMPTS(PROMPT1,PROMPT2); (*erase*)
   CHARTYPE(10);
```

```
    END; (*INFORM*)

BEGIN      (* SETUPCONDITIONS *)
  REPEAT
    NEWLABELS(1); (* display *)
    NEWLABELS(2); (* display *)
    ZERO:=FALSE;
    CONCSTR:='sodium carbonate';
    SELECTCONC(S1CONC,CONCSTR);
    IF NOT QUIT THEN
     BEGIN
       NEWLABELS(1);(*erase  'sodium carb' *)
       REALSTR(S1CONC,CONCSTR,3,5);
       SALTMOLARITY(CONCAT(CONCSTR,'M'));
       CONCSTR:='sodium bicarbonate';
       SELECTCONC(S2CONC,CONCSTR);
       IF NOT QUIT THEN
         BEGIN
           NEWLABELS(2);(*erase 'sodium bicarb' *)
           REALSTR(S2CONC,CONCSTR,3,5);
          TITRMOLARITY(CONCAT(CONCSTR,'M'));
          IF ((S1CONC=0.0) AND (S2CONC=0.0)) THEN
            BEGIN
              ZERO:=TRUE;
              SALTMOLARITY('     '); (* delete molarity of salt1 *)
              TITRMOLARITY('     '); (* delete molarity of salt2*)
              INFORM;
            END;
         END;
     END;
  UNTIL ((NOT ZERO) OR QUIT);
  INFLASK:=ASALT;
  IF NOT QUIT THEN SELECTVOL(FLASKVOL) ELSE ERASELABELS;
  IF NOT QUIT THEN
    BEGIN
      REALSTR(FLASKVOL,CONCSTR,2,6);
      SALTDISP(CONCSTR);
      CONCSTR:='hydrochloric acid';
      SELECTCONC(TCONC,CONCSTR);
   END;
  IF NOT QUIT THEN
   BEGIN
     REALSTR(TCONC,CONCSTR,3,5);
     ACIDDISP(CONCAT(CONCSTR,'M'));
   END;
END;  (* SETUPCONDITIONS *)


(*$I :MIXTURECAL*)
(*$I :MIXTURE2*)
(*$I :MIXTURE3*)
```

```
(*$I :MIXTURECAL*)
(************************************************************)
PROCEDURE CALCULPH(ANYSALT1,ANYSALT2,SALTVOL,ACIDVOL,ACIDCONC: REAL;
                                                    VAR PH: REAL);
(************************************************************)
(*ANYSALT1 & 2 - initial moles of both salts *)
VAR
 WACID,SACID,ACIDMOL, TOTALVOL,
 SALT1CONC, SALT2CONC, EXCESS  : REAL;
 A,B,C,D,APPROX :REAL;
 FIRST : BOOLEAN;

   (*=======================================================*)
   PROCEDURE NEWTONCUBIC(VAR PH :REAL);
   (*=======================================================*)
   (*iterates cubic equation -  requires global APPROX and constants A,B,C,D*)
   VAR
    COUNT :INTEGER;
    NEWTONX,ERROR,ONEPER, GUESS:  REAL;
    SOLN: BOOLEAN;

      (*----------------------------------------------------*)
      FUNCTION EQUATION(X : REAL):REAL;
      (*----------------------------------------------------*)
      BEGIN
       EQUATION:=D+X*(C+X*(B+A*X));
      END; (*EQUATION*)

      (*----------------------------------------------------*)
      FUNCTION DERIV(X : REAL):REAL;
      (*----------------------------------------------------*)
      BEGIN
       DERIV:=C+X*(3*A*X+2*B);
      END;

   BEGIN (*NEWTONCUBIC *)
    COUNT:=0;
    GUESS:=APPROX;
    SOLN:=FALSE;
    REPEAT
       COUNT:=COUNT+1;
       NEWTONX:=APPROX-(EQUATION(APPROX)/DERIV(APPROX));
       ERROR:=ABS(APPROX-NEWTONX);
       ONEPER:=NEWTONX * 0.01;
       IF (ERROR<ONEPER) THEN  SOLN:=TRUE  ELSE  APPROX:=NEWTONX;
    UNTIL ((COUNT>20) OR (SOLN));
    IF (NEWTONX<0.0) THEN
      BEGIN(* If negative root, make another guess & iterate until positive soln found*)
        APPROX:=GUESS*10;
        NEWTONCUBIC(PH);
      END
       ELSE PH:=-1*LOG(NEWTONX);
   END; (*NEWTONCUBIC *)
```

```
(*==============================================================*)
PROCEDURE QUADRATIC(WACID : REAL;  VAR H,PH  : REAL);
(*==============================================================*)
(* Soln of salt of weak acid/strong base titrated with s.acid. H+ results from excess of
strong & hydrolysis of salt is often insignificant *)
VAR  HYDROL  : REAL;
BEGIN
   HYDROL :=SQRT(K1 *WACID); (* H+ from hydrolysis of weak acid *)
   H:=H + HYDROL;
   PH :=-1 *LOG(H);
END;  (* QUADRATIC *)

(*==============================================================*)
PROCEDURE APPROX1(ACIDCONC : REAL;   VAR PH  : REAL);
(*==============================================================*)
(* Eqn 28 -simplified quartic eqn 26 assuming soln fairly acidic*)
BEGIN (*APPROX1*)
  A :=1 /K1 ;
  B :=1 ;
  C :=K2-ACIDCONC;
  D :=-2.0*K2*ACIDCONC;
  IF K1 >=1.0 THEN APPROX :=ACIDCONC ELSE APPROX :=SQRT(K1 *ACIDCONC);
  NEWTONCUBIC(PH);
END;  (*APPROXCUB1*)

(*==============================================================*)
PROCEDURE APPROX2(ACIDCONC,SALTCONC :REAL;  VAR PH :REAL);
(*==============================================================*)
(* EQN 36 IF ACIDIC & EQN 43 IF BASIC - SIMPLIFIED QUARTIC EQN 34 *)
BEGIN
  APPROX :=K1 *ACIDCONC/SALTCONC;
  IF K1 >=1.0 THEN APPROX :=0.01 *APPROX;
  IF APPROX>=10E-7 THEN
   BEGIN
     A :=1 /K1 ;
     B :=(SALTCONC/K1)+1 ;            (*EQN. 36 *)
     C :=K2-ACIDCONC;
     D :=-1.0*K2*(SALTCONC+2*ACIDCONC);
   END

  ELSE
   BEGIN
      A :=SALTCONC/(K1 *K2);
      B :=(KW /K1)+ACIDCONC;          (* EQN. 43 *)
      B :=-1 *B /K2;
      C :=-1 *(SALTCONC+(2*ACIDCONC)+KW /K2);
      D :=-KW;
    END;
 NEWTONCUBIC(PH);
END;

(*==============================================================*)
PROCEDURE APPROX3(SALTCONC :REAL;    VAR PH :REAL);
(*==============================================================*)
 (* EQN 51 IF ACIDIC & EQN 56 IF BASIC - SIMPLIFIED EQN 49 *)
BEGIN
```

```
 APPROX:=SQRT(K1*K2);
 IF APPROX>=1.0E-7 THEN
  BEGIN
    A:=1/K1;
    B:=(SALTCONC/K1) + 1;       (* EQN 51 *)
    C:=K2;
    D:=-1.0*(K2*SALTCONC);
  END
  ELSE
    BEGIN
      A:=SALTCONC/(KW*K1);
      B:=-1.0/K1;         (* EQN. 56 *)
      C:=(K2*SALTCONC/KW)+1;
      C:=-1.0*C;
      D:=-1.0*K2;
    END;
 NEWTONCUBIC(PH);
END; (*APPROXCUB3*)

(*==============================================================*)
PROCEDURE APPROX4(SALT1,SALT2:REAL; VAR PH: REAL);
(*==============================================================*)
(*Eqn 62 if acidic & eqn 70 if basic - simplified quartic eqn 60*)
BEGIN
 APPROX:=K2*SALT1/SALT2;
 IF APPROX>=1.0E-7 THEN
   BEGIN
     A:=1/K1;
     B:=((SALT1+2*SALT2)/K1)+1;   (*EQ62*)
     C:=K2+SALT2;
     D:=-1.0*(K2*SALT1);
   END
  ELSE
  BEGIN
     A:=(SALT1+2*SALT2)/K1;
     B:=SALT2-(KW/K1);   (*EQN 70*)
     C:=(K2*SALT1)+KW;
     C:=-1.0*C;
     D:=-1.0*KW*K2;
  END;
 NEWTONCUBIC(PH);
END; (*APPROXCUB4*)

(*==============================================================*)
PROCEDURE INITCALC;
(*==============================================================*)
(* passes vol of acid & base but returns moles of acid & base also returns conc of
species in excess if acid>base then excess will be +ve  otherwise it  will be -ve *)
CONST DIFF=0.000001;
BEGIN
    TOTALVOL:=SALTVOL+ACIDVOL;
    ACIDMOL:=ACIDVOL*ACIDCONC;
    SALT1CONC:=ANYSALT1/TOTALVOL;
    SALT2CONC:=ANYSALT2/TOTALVOL;
    EXCESS:=ANYSALT1-ACIDMOL;
    IF ABS(EXCESS)<DIFF THEN EXCESS:=0.0;
```

```
          FIRST :=EXCESS>=0.0;
          IF NOT FIRST THEN
           BEGIN   (* past 1st end pt*)
              EXCESS :=(ANYSALT2 + 2*ANYSALT1)-ACIDMOL;
              IF ABS(EXCESS)<DIFF THEN EXCESS :=0.0;
           END;
     END; (* INITCALC *)
BEGIN  (* CALCPH *)
  INITCALC;  (* vol. of anyacid & anybase converted into moles *)
  IF FIRST THEN          (*salt1>titrant ie. before lst end pt *)
   BEGIN               (* or start of titration. acid=0      *)
     IF EXCESS=0.0 THEN  (* or 1st end pt *)
        BEGIN
         SALT2CONC :=(ACIDMOL+ANYSALT2)/TOTALVOL;
         APPROX3(SALT2CONC,PH);
        END
        ELSE
        BEGIN
         SALT1CONC :=EXCESS/TOTALVOL;
         SALT2CONC :=(ANYSALT2 + ACIDMOL)/TOTALVOL;
         APPROX4(SALT2CONC,SALT1CONC,PH);
        END;
   END
   ELSE
   BEGIN
     IF EXCESS>0.0 THEN
      BEGIN
        SALT2CONC :=EXCESS/TOTALVOL; (*between 1st - 2nd endpt *)
        WACID :=(ACIDMOL-ANYSALT1)/TOTALVOL ;
        APPROX2(WACID,SALT2CONC,PH);
      END
     ELSE  IF EXCESS=0.0 THEN        (* 2nd end pt *)
      BEGIN
        WACID :=(ANYSALT1+ANYSALT2)/TOTALVOL;
        APPROX1(WACID,PH);
      END
      ELSE                   (*Past 2nd end pt *)
       BEGIN
         WACID :=ANYSALT1+ANYSALT2;
         SACID :=ACIDMOL-(2*ANYSALT1+ANYSALT2);
         WACID :=WACID/TOTALVOL;
         SACID :=SACID/TOTALVOL;
         QUADRATIC2(WACID,SACID,PH);
       END;
   END; (*ELSE*)
END; (* CALCULPH *)
```

```
(* MIXTURE2.text - included in MIXTURE   *)
(*****************************************************************)
PROCEDURE TITRATE;
(*****************************************************************)
VAR
   SALTCOL,ACIDCOL,                          (* soln colours during titration   *)
   SOLNCOL : SCREENCOLOR;                    (* current colour of soln in flask  *)
   SPACEPR,                                  (* flag to indicate space bar pressed *)
   SELECTCHANGE: boolean;        (* flag to indicate change in titrant increment vol. *)
   INCR,                                     (* current titrant increment vol.  *)
   BURVOL,                                   (* total vol. added from burette (titrant)  *)
   SALTVOL,TITRVOL,                          (* total vol. of salt & titrant in flask  *)
   ENDPT1,ENDPT2,                            (* vol. of titrant required to reach endpt  *)
   SMOL1,SMOL2,                              (* initial moles of both salts *)
   PH,                                       (* current pH of soln *)
   NEXTVOL,                                  (*nextvol required for graphing  *)
   XTRAVOL,                                  (* vol. titrant not yet shown to fill flask *)
   PHRATIO : REAL;  (*ratio between no. pixels on pHscale & pH range  plotted on scale*)
   INTENDPT,       (* integer value of endpt*100 required for indicator colour change  *)
   RTSIDE,LTSIDE,                            (* current x-coord. of flask being filled  *)
   OLDLEVEL,INCREASE,                        (* current & increase in level of soln  *)
   OLDX,OLDY,                                (* current coord of pH graph *)
   INDEX :INTEGER;                           (* required for graphing pH curve *)

   NEXTCOL,GRAPHCOL:SCREENCOLOR;
   PENCHANGE,TINT, INDICATOR:BOOLEAN;
   UPPER,LOWER:INTEGER;
   UPPERPH,LOWERPH:REAL;

   (*===========================================================*)
   PROCEDURE INITCONDITIONS(VAR ENDPT1 :REAL);
   (*===========================================================*)
   CONST  PHSCALE=100.0;  (* No. pixels on pH scale*)
          PHRANGE=14.0;  (* pH 0 - 14 equally spaced on pHscale*)
   VAR  VOLSTR: SHORTSTR;
        X,Y:INTEGER;
   BEGIN
     X:=XCON+8;
     Y:=YCON+110;
     BURVOL:=0.0;
     SALTVOL:=FLASKVOL;
     TITRVOL:=BURVOL;
     SMOL1:=SALTVOL*SALT1CONC;
     SMOL2:=SALTVOL*SALT2CONC;
     ENDPT1:=SALTVOL*SALT1CONC/TITRCONC;
     ENDPT2:=SALTVOL*SALT2CONC/TITRCONC;
     ENDPT2:=ENDPT2 + 2*ENDPT1;
     WSTAT(X,Y,'pH vs vol.acid');
     IF ENDPT1<320.0 THEN INTENDPT:=ROUND(ENDPT1 * 100)
       ELSE INTENDPT:=32000;(* intendpt required to change indicator-due to vol of
                  flask, if endpt>320 then titration will not reach  endpt *)
      PHRATIO:=(PHSCALE/PHRANGE);  (* pH increm. per pixel *)
      FILLRATE:=2;  (*determines rate at which flask filled*)
      REALSTR(TITRVOL,VOLSTR,2,6);
      TITRDISP(VOLSTR);          (* Display ACID volume *)
      IF SMOL2=0 THEN  PH:=-LOG(SQRT(KW*K2/SALT1CONC)) (*only carbonate *)
```

```
     ELSE CALCULPH(SMOL1,SMOL2,SALTVOL,TITRVOL,TITRCONC,PH);
 END; (*INITCONDITIONS*)
(*==============================================================*)
PROCEDURE INITGRAPH;
(*==============================================================*)
BEGIN
 OLDX:=XCON;OLDY:=PHPTS[0]+YCON;
 INDEX:=0; NEXTVOL:=VOLPTS[1];
END;

(*==============================================================*)
PROCEDURE INITLEVEL(VAR RIGHTS,LEFTS,TOPLEVEL:INTEGER);
(*==============================================================*)
(* initializes coord of sides of flask & top of flask as well as level of soln in flask   *)
CONST WIDTH=2; (* indent soln from sides of flask *)
BEGIN
 FLASKTOP:=FLASKY+ (3*FLASKSIZ)DIV 4;  (*y-coord. of top sloping sides of flask*)
 NECKTOP:=FLASKY+FLASKSIZE;   (*y-coord. of very top of flask *)
 LEFTS:=FLASKX-(FLASKSIZ DIV 2)+WIDTH; (*calc. coord of sides*)
 RIGHTS:=FLASKX+(FLASKSIZ DIV 2)-WIDTH; (*of flask given midpt. of base*)
 TOPLEVEL:=FLASKY+1;              (* base of flask= Flasky  *)
 INCREASE:=10; (*depth of soln to be initially placed in flask*)
 XTRAVOL:=0.0;            (*initialize increment in titrant *)
END;(*INITLEVEL*)

(*==============================================================*)
PROCEDURE SETUPARRAYS(VAR VOLPTS:REALPTS; VAR PHPTS: INTPTS);
(*==============================================================*)
(* calculate pH value for volscale no. points. Volume calculated is twice required to
reach end point if  monoprotic  and three times if diprotic *)
CONST   MIN=0.01; MAX=10.00;   (*min & max value of incr. of titrant *)
        X=195; Y=10;      (*coord . for display of increment selected*)
VAR   I:INTEGER;
      VOLRATIO:REAL;  (*ratio of vol. of titrant plotted to no. pixels onx- axis*)
      INCSTR:STRING;  PHSTR:SHORTSTR;
      CH:CHAR;

   (*-----------------------------------------------------------*)
   PROCEDURE INITARRAYS;
   (*-----------------------------------------------------------*)
   BEGIN
      PHPTS[0]:=ROUND(PH*PHRATIO);
      VOLRATIO:=(ENDPT2*1.4)/VOLSCALE;     (* vol.incr. for each pixel*)
      FOR I:=1 TO VOLSCALE DO VOLPTS[I]:=VOLRATIO*I; (*total vol.at point 'I'*);
   END; (* INITARRAYS  *)

   (*-----------------------------------------------------------*)
   PROCEDURE INFORMPH;
   (*-----------------------------------------------------------*)
   BEGIN
      WSTAT(3,10,CONCAT('Initial pH is ',PHSTR));
      WSTAT(65,0,'Press <SPACE BAR> to continue');
   END; (*INFORMPH*)
```

```
(*-----------------------------------------------------------------------*)
 PROCEDURE PLEASEWAIT;
(*-----------------------------------------------------------------------*)
BEGIN
   WSTAT(10,5,'PREPARING SOLUTIONS ......');
END;

(*-----------------------------------------------------------------------*)
 PROCEDURE CYCLE;
(*-----------------------------------------------------------------------*)
 BEGIN
   IF NOT AGAIN THEN  WHILE ((I<VOLSCALE) AND (NOT KEYIN)) DO
      BEGIN
        I:=I+1;
        CALCULPH(SMOL1,SMOL2,FLASKVOL,VOLPTS[I],TITRCONC,PH);
        PHPTS[I]:=ROUND(PHRATIO*PH);
      END;(* WHILE *)
  END; (* CYCLE *)

BEGIN(*SETUPARRAYS*)
   IF (NOT AGAIN) THEN INITARRAYS;
   I:=0;
  INCRPROMPT;
  CYCLE;
  INRANGERESPONSE(INCR,INCSTR,MIN,MAX,X,Y); (*get increment*)
  CHARTYPE(6);INCRPROMPT;CHARTYPE(10);(*erase prompt for incr.*)
  IF QUIT THEN EXIT(SETUPARRAYS);

   REALSTR(PH,PHSTR,2,5);
   INFORMPH;        (*Inform initial pH*)
   CYCLE;
   GETACHAR(CH,[SPACE,'Q']);
   DISPLAYPH(PHSTR); (* display initial pH*)
   CHARTYPE(6);INFORMPH;CHARTYPE(10);
   QUIT:=CH='Q';
   IF QUIT THEN EXIT(SETUPARRAYS);

  IF ((I<VOLSCALE) AND (NOT AGAIN))THEN
   BEGIN
    PLEASEWAIT;  (*Display prompt*)
    REPEAT
      CYCLE;
      IF KEYIN THEN READ(CH);
    UNTIL I=VOLSCALE;
    CHARTYPE(6);PLEASEWAIT;CHARTYPE(10); (*erase prompt*)
   END;
END;(* SETUPARRAYS *)

(*======================================================================*)
PROCEDURE SLIGHTCHANGE(CURRENTL:INTEGER);
(*======================================================================*)
(* slight traces of other colour degree is an integer 1-10 indicating degree of second
colour  on top of existing soln col*)
VAR  Y,RSIDE,LSIDE,SHADE,DEGREE:INTEGER;
     BITCOL:SCREENCOLOR;
```

```
    (*----------------------------------------------------------------*)
    PROCEDURE DRAWDOTS(START,FIN,YY:INTEGER);
    (*----------------------------------------------------------------*)
    VAR LENGTH :INTEGER;
     BEGIN
        LENGTH:=3;
        FIN:=FIN-LENGTH;
        START:=START+SHADE;
        WHILE START<FIN DO
          BEGIN
            MOVECOL(START,YY,BITCOL);
            START:=START+LENGTH;
            MOVECOL(START,YY,NONE);
            START:=START+DEGREE;
          END;
     END; (*DRAWDOTS*)

BEGIN (*SLIGHTCHANGE*)
   IF (UPPERPH-LOWERPH)=0.0 THEN EXIT(SLIGHTCHANGE); (*precaution*)
   INITLEVEL(RSIDE,LSIDE,Y);
   DEGREE:=ROUND((PH-LOWERPH)/(UPPERPH-LOWERPH)*11);
   IF SOLNCOL=ACIDCOL THEN BITCOL:=SALTCOL ELSE BITCOL:=ACIDCOL;
   IF INDNUM='3' THEN DEGREE:=DEGREE+2;
   SHADE:=(DEGREE+1) DIV 3;
   IF SHADE=0 THEN SHADE:=1;(*intense 1-3*)
   CASE DEGREE OF
    1,5,7: Y:=Y+1;
    2,3  : LSIDE:=LSIDE+2;
    6,9  : BEGIN
             Y:=Y+3;
             RSIDE:=RSIDE-1;
             LSIDE:=LSIDE+4;
           END;
    END;(*CASEF*)
   IF Y<FLASKTOP THEN
     REPEAT
       DRAWDOTS(LSIDE,RSIDE,Y);
       Y:=Y+2*SHADE;
       RSIDE:=RSIDE-SHADE;
       LSIDE:=LSIDE+SHADE;
     UNTIL(Y>=CURRENTL-1)OR(Y>=FLASKTOP);
     WHILE(Y<NECKTOP) AND (Y<CURRENTL-1) DO
      BEGIN
       DRAWDOTS(LSIDE,RSIDE,Y);
       Y:=Y+2*SHADE;
      END;
   TINT:=FALSE;
END; (*SLIGHTCHANGE*)

(*================================================================*)
PROCEDURE ADDMORE;
(*================================================================*)
(* Increment vol of titrant & calc new pH; Display new pH & new volume of titrant*)
CONST BLANK='      ';
VAR VOL: INTEGER; PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
```

```
   BURVOL:=BURVOL+INCR;  (* calculate total vol. of titrant   *)
   VOL:=ROUND(BURVOL*100); (* this prevents build up of floating *)
   BURVOL:=VOL/100.0;      (* point errors *)
   REALSTR(BURVOL,VOLSTR,2,6);    (* convert vol. to string   *)
   CASE INFLASK OF         (* display vol of titrant on screen *)
    ACID,
    BASE : BEGIN           (* as either vol. of base or acid   *)
             SALTVOL:=BURVOL;
             SALTDISP(BLANK);
             SALTDISP(VOLSTR);
           END;
    ASALT: BEGIN
             TITRVOL:=BURVOL;
             TITRDISP(BLANK);
             TITRDISP(VOLSTR);
           END
   END; (* CASE *)
   CALCULPH(SMOL1,SMOL2,SALTVOL,TITRVOL,TITRCONC,PH);
   DISPLAYPH(BLANK);                 (* Erase old pH     *)
   REALSTR(PH,PHSTR,2,5);              (* convert to string *)
   DISPLAYPH(PHSTR);                 (* Display new pH   *)
 END;  (* ADDMORE *)
```

```
(* MIXTURE3 is included in MIXTURE    *)
 (*===============================================================*)
 PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
 (*===============================================================*)
 (*Change colour of soln & change label of soln in flask*)
 VAR CURRENTL,DEPTH : INTEGER;
 BEGIN
  SOLNCOL:=NEWCOLOR;
  CURRENTL:=OLDLEVEL;
  INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
  DEPTH:=CURRENTL-OLDLEVEL;
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
  IF CURRENTL>OLDLEVEL THEN (* colour change with soln in neckof flask*)
    BEGIN
      DEPTH:=CURRENTL-OLDLEVEL;
       FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
    END;
 END; (*CHANGECOL*)

 (*===============================================================*)
 PROCEDURE CHECKINDICATOR;
 (*===============================================================*)
 VAR TEMPCOL:SCREENCOLOR;
 BEGIN
   IF ((SOLNCOL=SALTCOL) AND (PH<=UPPERPH)) THEN
    BEGIN
      IF PH<=LOWERPH THEN
       BEGIN
        SOLNCOL:=ACIDCOL;
        INDICATOR:=TRUE;
       END
      ELSE TINT:=TRUE;
    END;

 IF INDICATOR THEN
   BEGIN
     IF ((SOLNCOL=BLACK2) OR (SOLNCOL=BLUE) OR (SOLNCOL=ORANGE))
      THEN TEMPCOL:=WHITE2 ELSE TEMPCOL:=WHITE1;
     IF GRAPHCOL<>TEMPCOL THEN
      BEGIN
        NEXTCOL:=TEMPCOL;
        PENCHANGE:=TRUE;
      END;
   END;
 END; (* CHECKINDICATOR*)

 (*===============================================================*)
 PROCEDURE GRAPH(VAR OLDX,OLDY,INDEX : INTEGER; VAR NEXTVOL: REAL);
 (*===============================================================*)
 (*To plot graph use values of pH already scaled & stored as integers in pHpts array.
 Plot all pts with corresponding titrant vol. of less than or equal to burette volume*)
 VAR EXACTPH, (*scaled integer pH calc. from current pH value *)
       X,Y : INTEGER; (*new X,Ycoordinate of graph *)
```

```
  (*----------------------------------------------------------------*)
  PROCEDURE SWAPPEN(NEWY:INTEGER);
  (*----------------------------------------------------------------*)
  BEGIN
    IF (NEWY<=UPPER) THEN
      BEGIN
        IF OLDY>UPPER THEN MOVETO(OLDX,UPPER);
        GRAPHCOL:=NEXTCOL;
        PENCOLOR(GRAPHCOL);
        PENCHANGE:=FALSE;
      END;
  END; (*SWAPPEN*)

BEGIN (*GRAPH*)
 MOVETO(OLDX,OLDY);
 PENCOLOR(GRAPHCOL);
 WHILE (BURVOL>=NEXTVOL) AND (INDEX<VOLSCALE) DO
  BEGIN
   INDEX:=INDEX+1;
   X:=INDEX+XCON; Y:=PHPTS[INDEX]+YCON;
   IF PENCHANGE THEN SWAPPEN(Y);
   MOVETO(X,Y);
   NEXTVOL:=VOLPTS[INDEX+1];
   OLDX:=X;
   OLDY:=Y;
  END;
 IF (INDEX<VOLSCALE) THEN
  BEGIN
    EXACTPH:=ROUND(PH*PHRATIO);
    EXACTPH:=EXACTPH+YCON;
    IF PENCHANGE THEN SWAPPEN(EXACTPH);
    MOVETO(OLDX,EXACTPH);
    OLDY:=EXACTPH;
  END;
 PENCOLOR(NONE);
END; (*GRAPH*)

(*================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*================================================================*)
(*Volume of solution in flask is only shown to increase when a suitable volume (say
5mL or more) has been released from burette.Therefore smaller increments are
summed until this volume is reached and then level of soln is shown to rise *)
VAR  EXTRA : INTEGER;
BEGIN
 XTRAVOL:=XTRAVOL+INCR; (*xtravol. is vol.titrant added that has not yet been
                                                    shown to fill flask*)
 IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled by an
                     extra amt.  Thisvalue must be even due to slope of flask*)
BEGIN
    EXTRA:=TRUNC(XTRAVOL/FILLRATE);
    IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
    FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
    XTRAVOL:=0;
  END;
END;(*CHECKLEVEL*)
```

```
(*=================================================================*)
PROCEDURE INITIND(INDNUM:CHAR; VAR LOWERPH,UPPERPH: REAL;
                                     VAR ACIDCOL,SALTCOL: SCREENCOLOR);
(*=================================================================*)
VAR INDSTR: STRING[14];

    (*------------------------------------------------------------*)
    PROCEDURE METHYLO;
     (*-----------------------------------------------------------*)
    BEGIN
      LOWERPH:=3.1;UPPERPH:=4.4;
      ACIDCOL:=VIOLET;SALTCOL:=ORANGE;
      INDSTR:='methyl orange';
    END;

    (*------------------------------------------------------------*)
    PROCEDURE METHYLR;
    (*------------------------------------------------------------*)
    BEGIN
      LOWERPH:=4.2;UPPERPH:=6.2;
      ACIDCOL:=VIOLET;SALTCOL:=ORANGE;
      INDSTR:='methyl red';
    END;

    (*------------------------------------------------------------*)
    PROCEDURE BROMOB;
    (*------------------------------------------------------------*)
    BEGIN
      LOWERPH:=6.0;UPPERPH:=7.6;
      ACIDCOL:=ORANGE;SALTCOL:=BLUE;
      INDSTR:='bromo.blue';
    END;

    (*------------------------------------------------------------*)
    PROCEDURE PHENOL;
    (*------------------------------------------------------------*)
    BEGIN
      LOWERPH:=8.3;UPPERPH:=10.0;
      ACIDCOL:=BLACK1;SALTCOL:=VIOLET;
      INDSTR:='phenolphth.';
    END;

    (*------------------------------------------------------------*)
    PROCEDURE HYPOTH;
    (*------------------------------------------------------------*)
    BEGIN
      IF INDNUM='6' THEN (*2nd end pt*)
        CALCULPH(SMOL1,SMOL2,SALTVOL,ENDPT2,TITRCONC,LOWERPH)
          ELSE   CALCULPH(SMOL1,SMOL2,SALTVOL,ENDPT1,TITRCONC,LOWERPH);
      UPPERPH:=LOWERPH;
      ACIDCOL:=ORANGE; SALTCOL:=BLUE;
      INDSTR:='Hypothetical';
    END; (*HYPOTH*)

 BEGIN (* INITIND *)
    CASE INDNUM OF
```

```
        '1': METHYLO;
        '2': METHYLR;
        '3': BROMOB;
        '4': PHENOL;
        '5',
        '6': HYPOTH;
       END; (*CASE*)
       WSTAT(XCON-10,YCON-14,INDSTR);
     IF NOT COLOUR THEN
       BEGIN
         ACIDCOL:=BLACK1;
         SALTCOL:=VIOLET;
       END;
   END; (* INITIND*)

   (*==============================================================*)
   PROCEDURE SHOWRANGE(VAR LOWERPH,UPPERPH:REAL;
           ACIDCOL,SALTCOL:SCREENCOLOR);
   (*==============================================================*)
   VAR  START,FIN : INTEGER;
      (*-----------------------------------------------------------*)
      PROCEDURE MIDWAY;
      (*-----------------------------------------------------------*)
      VAR CENTRY,DIST,LINE,GAP:INTEGER;

         (*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)
         PROCEDURE DRAWDOTS(X1,X2,Y:INTEGER; COL:SCREENCOL);
         (*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - *)
         VAR LENGTH,ON,OFF,INDENT:INTEGER;
         BEGIN
           LENGTH:=2;
           X2:=X2-LENGTH;
           CASE LINE OF
                    0:   BEGIN ON:=1; OFF:=2; END;
                  1,3:   BEGIN ON:=1; OFF:=3; END;
              2,4,5,6 :  BEGIN ON:=2; OFF:=2; END;
         7,8,9,10,11,12: BEGIN ON:=3; OFF:=2; END;
             13,14,15:   BEGIN ON:=3; OFF:=1; END;
              END; (*CASE*)
           CASE LINE OF
             1,3,8,10,12:  INDENT:=0;
                 0,2,4,6:  INDENT:=1;
                     7,9:  INDENT:=2;
             5,11,13,15:   INDENT:=3;
             END; (*CASE*)
           IF ((LINE=1) OR (LINE=3)) THEN
             BEGIN
               IF ((COL=SALTCOL) AND (ACIDCOL<>BLACK1)) THEN COL:=ACIDCOL
                ELSE COL:=SALTCOL;
             END;
           X1:=X1+INDENT*LENGTH;
           WHILE X1<X2 DO
             BEGIN
               MOVECOL(X1,Y,COL);
               X1:=X1+ON*LENGTH;
               MOVECOL(X1,Y,NONE);
```

```
                X1:=X1+OFF*LENGTH;
              END;
          END; (*DRAWDOTS*)

      BEGIN (*MIDWAY*)
        GAP:=2;
        DIST:=2; LINE:=0;
        IF ACIDCOL=BLACK THEN CENTRY:=LOWER+1
             ELSE CENTRY:=(UPPER+LOWER+1) DIV 2;
        DRAWDOTS(START,FIN,CENTRY,SALTCOL);
        WHILE(CENTRY+DIST)<UPPER DO
          BEGIN
            LINE:=LINE+1;
            IF ACIDCOL<>BLACK1 THEN
            BEGIN
              DRAWDOTS(START,FIN,CENTRY+DIST,SALTCOL);
              DRAWDOTS(START,FIN,CENTRY-DIST,ACIDCOL);
            END
             ELSE  DRAWDOTS(START,FIN,CENTRY+DIST,SALTCOL);
           DIST:=DIST+GAP;
          END
      END; (*MIDWAY*)

  BEGIN (*SHOWRANGE*)
     LOWER:=ROUND(PHRATIO*LOWERPH);
     UPPER:=ROUND(PHRATIO*UPPERPH);
     LOWER:=YCON+LOWER;
     UPPER:=YCON+UPPER;
     START:=XCON+2;
     FIN:=XCON+VOLSCALE;
     VIEWPORT(START,FIN,YCON+2,LOWER);
     FILLSCREEN(ACIDCOL);
     VIEWPORT(START,FIN,UPPER,YCON+100);
     FILLSCREEN(SALTCOL);
     VIEWPORT(XMIN,XMAX,YMIN,YMAX);
     MIDWAY;
     DRAWLINE(START,LOWER,FIN,LOWER,WHITE2);
     DRAWLINE(START,UPPER,FIN,UPPER,WHITE2);
  END; (*SHOWRANGE*)

  (*==============================================================*)
  PROCEDURE INITCOLOURS;
  (*==============================================================*)
  BEGIN
     IF PH>=LOWERPH THEN SOLNCOL:=SALTCOL ELSE SOLNCOL:=ACIDCOL;
     TINT:=((PH<UPPERPH) AND (PH>LOWERPH));
     IF ((SOLNCOL=BLACK2) OR (SOLNCOL=BLUE) OR (SOLNCOL=ORANGE))
        THEN GRAPHCOL:=WHITE2  ELSE GRAPHCOLL:=WHITE1;
     NEXTCOL:=GRAPHCOL;
  END; (*INITCOLOURS*)
```

```
BEGIN   (* TITRATE *)
   INITCONDITIONS(ENDPT1);
   INITIND(INDNUM,LOWERPH,UPPERPH,ACIDCOL,SALTCOL);
   INITCOLOURS;
   INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
   SHOWRANGE(LOWERPH,UPPERPH,ACIDCOL,SALTCOL);
   FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
   IF TINT THEN SLIGHTCHANGE(OLDLEVEL);
   SETUPARRAYS(VOLPTS,PHPTS);
   INITGRAPH;
   PENCHANGE:=FALSE;
   IF NOT QUIT THEN
      BEGIN
        REQUEST;        (*Display prompt to press space bar*)
        SELECTCHANGE:=FALSE;
         REPEAT
           CHECKKEY(SPACEPR,SELECTCHANGE);
           IF SPACEPR THEN
             BEGIN
               INDICATOR:=FALSE;
               MOVEDROP(INCR,OLDLEVEL);
               ADDMORE;
               CHECKINDICATOR;
               IF INDICATOR THEN CHANGECOL(SOLNCOL)
                 ELSE IF TINT THEN SLIGHTCHANGE(OLDLEVEL);
               CHECKLEVEL(XTRAVOL,INCR);
               GRAPH(OLDX,OLDY,INDEX,NEXTVOL);
             END;
           IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
          UNTIL QUIT;
         CHARTYPE(0);
         REQUEST;           (*erase prompt*)
         CHARTYPE(10);
         AGAIN:=(BURVOL>0); (* only give option to repeatif titration has commenced*)
      END;
END; (* TITRATE *)

(*************************************************************)
PROCEDURE GETINDICATOR(VAR INDNUM:CHAR);
(*************************************************************)
CONST X=0; DOTS='    .....(';
VAR   Y: INTEGER;
BEGIN
   Y:=0;
   PAGE(OUTPUT);
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
   WRITE(AT(X+7,Y),'INDICATORS AVAILABLE:'); Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
   WRITE(AT(X,Y),'Methyl orange (3.1-4.4)   ',DOTS,'1)'); Y:=Y+2;
   WRITE(AT(X,Y),'Methyl red (4.2-6.2)      ',DOTS,'2)'); Y:=Y+2;
   WRITE(AT(X,Y),'Bromothymol blue (6.0-7.6)',DOTS,'3)');Y:=Y+2;
   WRITE(AT(X,Y),'Phenolphthalein (8.3-10.0)',DOTS,'4)'); Y:=Y+2;
   WRITE(AT(X,Y),'Hypothetical(1st equiv.pt)',DOTS,'5)'); Y:=Y+2;
   WRITE(AT(X,Y),'Hypothetical(2nd equiv.pt)',DOTS,'6)');Y:=Y+3;
   WRITE(AT(X+7,Y),'SELECT INDICATOR  ',DOTS,'  )');
   GETTEXTCHAR(37,Y,INDNUM,['1'..'6','Q']);
```

```
  QUIT :=INDNUM='Q';
  IF QUIT THEN AGAIN :=FALSE;
  PAGE(OUTPUT);
END; (*GETINDIC*)

(****************************************************************)
PROCEDURE STARTAGAIN;
(****************************************************************)

  (*==========================================================*)
  PROCEDURE CLEARVALUES;
  (*==========================================================*)
  (* erase all values plus flask *)
  CONST  BLANK='     ';
  BEGIN
    DISPLAYPH(BLANK);      (* delete pH *)
    TITRDISP(BLANK);      (* delete vol. of SALT *)
    IF NOT AGAIN THEN
      BEGIN
        SALTMOLARITY(BLANK); (*delete molarity of salt1 *)
        TITRMOLARITY(BLANK); (*delete molarity of salt2*)
        SALTDISP(BLANK);       (* delete vol. of salt*)
        ACIDDISP(BLANK);      (* delete molarity of acid*)
      END;
    FILLBOX(10,110,25,125,BLACK1);  (* erase flask  *)
  END; (*CLEARVALUES*)

  (*======================================================*)
  PROCEDURE CHECKAGAIN;
  (*======================================================*)
  CONST  X=0;
         DOTS='   .....(';
  VAR Y :INTEGER;
      CH :CHAR;
  BEGIN
   Y :=6; PAGE(OUTPUT);
   WRITE(AT(X,Y),'Repeat previous titration with ');Y :=Y+1;
   WRITE(AT(X+7,Y),'same indicator      ',DOTS,'R)');Y :=Y+1;
   WRITE(AT(X+7,Y),'different indicator ',DOTS,'D)');Y :=Y+3;
   WRITE(AT(X,Y),'Select different titration ',DOTS,'S)'); Y :=Y+3;
   WRITE(AT(X,Y),'Quit - back to main menu   ',DOTS,'Q)');Y :=Y+3;
   WRITE(AT(X+10,Y),'SELECT  OPTION  ',DOTS,'  )');
   GETTEXTCH(X+37,Y,CH,['R','D','S','Q']);
   IF AGAIN THEN AGAIN :=((CH='R') OR (CH='D'));
   NEWIND :=NOT((AGAIN) AND (CH='R'));
   QUIT :=CH='Q';  (* resets 'quit'  *)
  END; (* CHECKAGAIN*)

BEGIN (* STARTAGAIN *)
  PAGE(OUTPUT);
  FILLBOX(150,270,24,164,BLACK1); (* erase pH graph *)
  CHECKAGAIN;
  CLEARVALUES;
END; (* STARTAGAIN *)
```

```
BEGIN    (* main *)
    SETCHAIN(':SALTMENU');
    AGAIN:=FALSE;
    QUIT:=FALSE;
    NEWIND:=TRUE;
    INITSCREEN;
    ERASEBOXES;
    ERASELABELS;
    SETCOLOUR;
    SELECTTYPE(TITRTYPE);
    WHILE (NOT QUIT) DO
      BEGIN
        DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE2);
        DRAWAXES(XCON,YCON,VOLSCALE,WHITE2);
        IF ((NOT QUIT) AND (NEWIND)) THEN GETINDICATOR(INDNUM);
        GRAFMODE;
        IF (NOT AGAIN) AND (NOT QUIT) THEN
          SETCONDITIONS(SALT1CONC,SALT2CONC,TITRCONC,INFLASK);
        IF NOT QUIT THEN TITRATE;
        TEXTMODE;
        STARTAGAIN;
      END; (* while *)
    BACKTOMENU;
END. (*SALTMIXTURE*)
```

```
(*$S++*)(*$R-*)(*$V-*)(*$I-*)

PROGRAM SALTASSIGN;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,SALTLIB;

VAR
  ACIDIC,
  COLOUR,                          (* is a colour monitor available    *)
  NEW,                             (* flag to select a new assignment  *)
  AGAIN : BOOLEAN;                 (* option to repeat titration       *)
  UNKNOWN: ACIDORBASE;             (* type of unknown soln             *)
  UNKNOWNC,                        (* concentration of unknown soln    *)
  STDSOLN :REAL;                   (* concentration of standard soln   *)
  SOLSTR: STRING[17];              (* string of type of unknown soln   *)
  SALTCONC,TITRCONC:REAL;          (* conc. of solns                   *)


(***********************************************************)
PROCEDURE GETCALCULATOR;
(***********************************************************)
CONST  TOP=10; INDENT=20;
 VAR  Y: INTEGER;
      NUM1,NUM2: REAL;
      ESCAPE:BOOLEAN;
      OP:CHAR;

   (*======================================================*)
   PROCEDURE LAYOUT;
   (*======================================================*)
   CONST STAR='*';
   VAR X,Y:INTEGER;
   BEGIN
    PAGE(OUTPUT);
    X:=10; Y:=2;
    WRITE(AT(X,Y),AROW(21,STAR));     Y:=Y+2;
    WRITE(AT(X,Y),' C A L C U L A T O R'); Y:=Y+2;
    WRITE(AT(X,Y),AROW(21,STAR));
    WRITE(AT(X,23),'<Q> QUIT    <C> CLEAR');
    X:=4; Y:=TOP;
    WRITE(AT(X,Y),'+');Y:=Y+2;
    WRITE(AT(X,Y),'-');Y:=Y+2;
    WRITE(AT(X,Y),'X');Y:=Y+2;
    WRITE(AT(X,Y),'/');
   END; (* LAYOUT*)

   (*======================================================*)
   PROCEDURE ENTERNUM(Y:INTEGER;  VAR NUM:REAL);
   (*======================================================*)
   CONST X=10;
   VAR S:SHORTSTR;
   BEGIN
    WRITE(AT(X,Y),'ENTER NUM:');
    GETRESPONSE(22,Y,S,8,['0'..'9','.','Q','C']);
    WRITE(AT(X,Y),'          ');
    QUIT:=((S='Q') OR (S='C'));
```

```
 ESCAPE:=S='Q';
 NUM:=RVALUE(S);
END; (* ENTERNUM *)

(*==============================================================*)
PROCEDURE ENTEROP(Y:INTEGER; VAR OP:CHAR);
(*==============================================================*)
BEGIN
 GOTOXY(INDENT,Y);
 GETACHAR(OP,['+','-','X','*','/','Q','C']);
 WRITE(OP);
 QUIT:=((OP='Q') OR (OP='C'));
 ESCAPE:=OP='Q';
END;

(*==============================================================*)
PROCEDURE CLEAR(A,B: INTEGER);
(*==============================================================*)
CONST BLANKL='         ';
VAR  J: INTEGER;
BEGIN
 FOR J:=A TO B DO WRITE(AT(INDENT,J),BLANKL);
END; (* CLEAR*)

(*==============================================================*)
PROCEDURE CALC(VAR NUM1,NUM2:REAL; OP:CHAR);
(*==============================================================*)
BEGIN
 CASE OP OF
   '+': NUM1 :=NUM1+NUM2;
   '-': NUM1 :=NUM1-NUM2;
 'X','*': NUM1 :=NUM1*NUM2;
   '/': NUM1 :=NUM1/NUM2;
  END; (*CASE*)
  WRITE(AT(INDENT,Y),NUM1:9:5);
 END; (* CALC *)

BEGIN (*CALCULATOR*)
 LAYOUT;
 REPEAT
  Y:=TOP;
  ENTERNUM(Y,NUM1);
  IF NOT QUIT THEN
   REPEAT
    Y:=Y+2;
    ENTEROP(Y,OP);
    IF NOT QUIT THEN
     BEGIN
      Y:=Y+2;
      ENTERNUM(Y,NUM2);
      CLEAR(TOP,Y);
      Y:=TOP;
     END;
    IF NOT QUIT THEN CALC(NUM1,NUM2,OP);
   UNTIL QUIT;
  CLEAR(TOP,Y);
```

```
  UNTIL ESCAPE;
  QUIT:=FALSE;
  PAGE(OUTPUT);
END; (*CALCULATOR*)

(****************************************************************)
PROCEDURE GETSPACEBAR;
(****************************************************************)
VAR CH:CHAR;
BEGIN
  WRITE(AT(23,23),'Press <SPACE BAR>');
  GETACHAR(CH,[SPACE,'Q']);
  QUIT:=CH='Q';
END;(* GETSPACEBAR*)

(****************************************************************)
PROCEDURE SELECTUNKNOWN;
(****************************************************************)
CONST STAR='*';
VAR  FIRSTNO:REAL;
     X,Y:INTEGER;
     ASSIGN:STRING[2];
     CH:CHAR;

   (*==========================================================*)
   PROCEDURE SELECTTYPE;
   (*==========================================================*)
   BEGIN
   CASE ASSIGN[1] OF
    '0', '1', '2': BEGIN
                  TITRTYPE:=ACIDSALT;
                  SOLSTR:='AMMONIUM CHLORIDE';
                  K1:=1.79E-5; (* Kb for ammonia*)
                END;
    '3','4', '5', '6': BEGIN
                     TITRTYPE:=BASESALT;
                     SOLSTR:='SODIUM ACETATE';
                     K1:=1.76E-5; (* Ka for acetic acid*)
                   END;
    '7','8','9': BEGIN
                 TITRTYPE:=DISALT;
                 SOLSTR:='SODIUM CARBONATE';
                 K1:=4.3E-7; K2:=5.6E-11;(* K values for carbonic *)
               END;
    END; (*CASE*)
    PAGE(OUTPUT);
    ACIDIC:=(TITRTYPE=ACIDSALT);
    UNKNOWN:=ASALT;
  END; (* SELECTTYPE *)

BEGIN (* SELECTUNKNOWN *)
 PAGE(OUTPUT);
 X:=0; Y:=6;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X,Y),'ENTER ASSIGNMENT NO.(1-99):'); Y:=Y+2;
 WRITE(AT(X,Y),'This no. will be given to you by'); Y:=Y+1;
```

```
WRITE(AT(X,Y),'your teacher.'); Y:=Y+2;
WRITE(AT(X,Y),AROW(40,STAR));
GETRESPONSE(X+35,Y-5,ASSIGN,2,NUMS+['Q']);
QUIT:=(POS('Q',ASSIGN)>0);
IF QUIT THEN
 BEGIN
  BACKTOMENU;
  EXIT(SELECTUNKNOWN);
 END;
IF LENGTH(ASSIGN)<2 THEN
  ASSIGN:=CONCAT('0',ASSIGN);
SELECTTYPE;
FIRSTNO:=(ORD(ASSIGN[1])-48)*7;
UNKNOWNC:=ORD(ASSIGN[2])-48;
UNKNOWNC:=(100+(92*UNKNOWNC)+FIRSTNO)/1000;

SALTCONC:=UNKNOWNC;

PAGE(OUTPUT);
Y:=4;
WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
WRITE(AT(X,Y),' YOUR ASSIGNMENT IS TO CALCULATE'); Y:=Y+2;
WRITE(AT(X,Y),' THE CONCENTRATION OF A SOLUTION'); Y:=Y+2;
WRITE(AT(X,Y),'   OF ',SOLSTR,'.'); Y:=Y+2;
WRITE(AT(X,Y),' CONCENTRATION WILL BE IN RANGE');Y:=Y+2;
WRITE(AT(X,Y),'        0.100-1.000M');Y:=Y+2;
WRITE(AT(X,Y),AROW(40,STAR));
WRITE(AT(10,22),'Press <SPACE BAR> to continue ');
GETACHAR(CH,[SPACE,'Q']);
QUIT:=CH='Q';
PAGE(OUTPUT);
END; (* SELECTUNKNOWN *)

(***********************************************************)
PROCEDURE STANDARDSOLN;
(***********************************************************)
CONST STAR='*';
      X=0;  BLANK='        ';
VAR  S:STRING; CH:CHAR;
     OK:BOOLEAN;
     Y, J,INTNUM : INTEGER;
BEGIN
 IF ACIDIC THEN S:='sodium hydroxide.' ELSE S:='hydrocholoric acid.';
 Y:=8;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X,Y),'Enter concentration of standard'); Y:=Y+2;
 WRITE(AT(X,Y),'solution of ',S);Y:=Y+2;
 WRITE(AT(X,Y),'(0.100-1.000): ');Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,STAR));
 S:='';
 REPEAT
  GETRESPONSE(20,Y-2,S,5,NUMS+['Q']);
  QUIT:=S='Q';
  IF NOT QUIT THEN
   BEGIN
    STDSOLN:=RVALUE(S);
```

```
      OK:=((STDSOLN>=0.10) AND (STDSOLN<=1.0));
      IF OK THEN INTNUM:=TRUNC(STDSOLN*100);
      IF NOT OK THEN WRITE(AT(34,Y),BLANK);
    END;
  UNTIL OK OR QUIT;
 TITRCONC:=STDSOLN;
 PAGE(OUTPUT);
END; (* STANDARDSOLN *)

(*****************************************************)
PROCEDURE GETCONDITIONS;
(*****************************************************)
VAR ASTR: SHORTSTR;
    PROMPT1,PROMPT2:STRING;

   (*=========================================================*)
   PROCEDURE SELECTVOL(VAR FLASKVOL: REAL);
   (*=========================================================*)
   CONST  MIN=10.0; MAX=50;        (*  range of volume*)
          X=215; Y=10;             (*  coord. to enter input *)
   BEGIN
     PROMPT1:=CONCAT('Select vol. of salt in flask:');
     PROMPT2:=' (10-50ml)';
     TWOPROMPTS(PROMPT1,PROMPT2);
     INRANGERESPONSE(FLASKVOL,ASTR,MIN,MAX,X,Y);
     CHARTYPE(6);
     TWOPROMPTS(PROMPT1,PROMPT2); (* ERASE *)
     CHARTYPE(10);
   END; (*SELECTVOL*)

   (*=========================================================*)
   PROCEDURE SELECT(VAR INFLASK: ACIDORBASE);
   (*=========================================================*)
   VAR CH:CHAR;
   BEGIN
     INFLASK:=ASALT;
     PROMPT1:='Salt solution is in flask';
     PROMPT2:='        Press <SPACE BAR> to continue';
     TWOPROMPTS(PROMPT1,PROMPT2);
     GETACHAR(CH,[SPACE,'Q']);
     QUIT:=CH='Q';
     CHARTYPE(6);
     TWOPROMPS(PROMPT1,PROMPT2);  (* ERASE *)
     CHARTYPE(10);
   END; (*SELECT*)

BEGIN
  SALTMOLARITY('?');
  REALSTR(TITRCONC,ASTR,3,5);
  TITRMOLARITY(ASTR);
  IF NOT QUIT THEN SELECT(INFLASK);
  IF NOT QUIT THEN SELECTVOL(FLASKVOL);
END;(* GETCONDITIONS *)
```

```
(************************************************************)
PROCEDURE SHOWTYPE;
(************************************************************)
VAR X,Y: INTEGER;
     S1 : STRING[18];
 BEGIN
   IF ACIDIC THEN S1 :='SODIUM HYDROXIDE'  ELSE S1 :='HYDROCHLORIC ACID';
   X:=210; Y:=135;
   WSTAT(X-(7*LENGTH(S1) DIV 2),Y,S1);
   Y:=Y-20;
   WSTAT(X,Y,'VS');
   Y:=Y-20;
   WSTAT(X-(7*LENGTH(SOLSTR) DIV 2),Y,SOLSTR);
   IF ACIDIC THEN S1 :='BASE' ELSE S1 :='ACID';
   Y:=Y-50;
   WSTAT(X-(7*8),Y,CONCAT(S1,' IN BURETTE'));
 END; (* SHOWTYPE *)

(*$I :SALTASS2*)
```

```
(* SALTASS2.text - included in SALTASSIGN*)
(***********************************************************)
PROCEDURE TITRATE;
(***********************************************************)
VAR
  SALTCOL,TITRCOL,                          (* soln colours during titration    *)
  SOLNCOL : SCREENCOLOR;                    (* current colour of soln in flask  *)
  SPACEPR,                                  (* flag to indicate space bar pressed *)
  SELECTCHANGE: boolean;    (* flag to indicate change in titrant increment volume *)
  INCR,                                     (* current titrant increment vol.   *)
  BURVOL,                                   (* total vol. added from burette (titrant) *)
  SALTVOL,TITRVOL,                          (* total vol. of salt & titrant in flask *)
  ENDPT1,                                   (* vol. of titrant required to reach endpt *)
  PH,                                       (* current pH of soln               *)
  XTRAVOL:REAL;                             (* vol. titrant not yet shown to fill flask *)
  RTSIDE,LTSIDE,                            (* current x-coord. of flask being filled *)
  OLDLEVEL,INCREASE,                        (* current & increase in level of soln *)
  OLDX,OLDY,                                (* current coord of pH graph        *)
  INDEX:INTEGER;                            (* required for graphing pH curve   *)


(***********************************************************)
PROCEDURE INITCONDITIONS(VAR ENDPT1:REAL);
(***********************************************************)
VAR  ASTR: SHORTSTR;
BEGIN
 IF COLOUR THEN
  BEGIN
   SALTCOL:=VIOLET; TITRCOL:=BLUE;
  END
  ELSE
   BEGIN
    SALTCOL:=WHITE1;
    TITRCOL:=BLACK1;
   END;
 BURVOL:=0.0;
  SALTVOL:=FLASKVOL;
  TITRVOL:=BURVOL;
  ENDPT1:=SALTVOL*SALTCONC/TITRCONC;
  SOLNCOL:=SALTCOL;

   FILLRATE:=2;  (*determines rate at which flask filled*)
   REALSTR(SALTVOL,ASTR,2,6);
   SALTDISP(ASTR);            (* Display salt volume*)
   REALSTR(TITRVOL,ASTR,2,6);
   TITRDISP(ASTR);           (* Display titrant volume *)
   CALCPH(SALTVOL,TITRVOL,SALTCONC,TITRCONC,PH); (*Calc. initpH*)
   REALSTR(PH,ASTR,2,5);
   DISPLAYPH(ASTR);
 END;  (*INITCONDITIONS*)

(***********************************************************)
PROCEDURE INITLEVEL(VAR RSIDE,LSIDE,TOPLEVEL:INTEGER);
(***********************************************************)
(* initializes coord of sides of flask & top of flask as well as level of soln in flask  *)
CONST WIDTH=2; (* indent soln from sides of flask *)
```

```
BEGIN
 FLASKTOP :=FLASKY+ (3*FLASKSIZ)DIV 4; (*y-coord. of top sloping sides of flask*)
 NECKTOP :=FLASKY+FLASKSIZE;  (*y-coord. of very top of flask *)
 LSIDE :=FLASKX-(FLASKSIZ DIV 2)+WIDTH; (*calc. coord of sides*)
 RSIDE :=FLASKX+(FLASKSIZ DIV 2)-WIDTH; (*of flask given midpt. of base*)
 TOPLEVEL :=FLASKY+1 ;              (* base of flask= Flasky  *)
 INCREASE :=10; (*depth of soln to be initially placed in flask*)
 XTRAVOL :=0.0;           (*initialize increment in titrant *)
END;(* INITLEVEL*)

(************************************************************)
PROCEDURE ADDMORE;
(************************************************************)
(* Increment vol. of titrant & calc. new pH - display new pH & new volume of titrant*)
CONST BLANK='     ';
VAR  VOL : INTEGER; PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
   BURVOL :=BURVOL+INCR;  (* calculate total vol. of titrant    *)
   VOL :=ROUND(BURVOL*100); (* this prevents build up of floating *)
   BURVOL :=VOL/100.0;     (* point errors *)
   REALSTR(BURVOL,VOLSTR,2,6);    (* convert vol.  to string  *)
   TITRVOL :=BURVOL;
   TITRDISP(BLANK);
   TITRDISP(VOLSTR);
   CALCPH(SALTVOL,TITRVOL,SALTCONC,TITRCONC,PH);
   DISPLAYPH(BLANK);                  (* Erase old pH     *)
   REALSTR(PH,PHSTR,2,5);               (* convert to string *)
   DISPLAYPH(PHSTR);                  (* Display new pH    *)
 END;  (* ADDMORE *)

(************************************************************)
PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
(************************************************************)
(*Change colour of soln & change label of soln in flask*)
VAR  CURRENTL,DEPTH : INTEGER;
BEGIN
   SOLNCOL :=NEWCOLOR;
   CURRENTL :=OLDLEVEL;
   INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
   DEPTH :=CURRENTL-OLDLEVEL;
   FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
   IF CURRENTL>OLDLEVEL THEN
     BEGIN
       DEPTH :=CURRENTL-OLDLEVEL;
       FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
     END;
END; (*CHANGECOL*)

(************************************************************)
PROCEDURE CHECKINDICATOR;
(************************************************************)
BEGIN
  IF ((SOLNCOL=SALTCOL) AND (BURVOL>=ENDPT1)) THEN CHANGECOL(TITRCOL);
END; (* CHECKINDICATOR*)
```

```
(*************************************************************)
PROCEDURE CHECKLEVEL(VAR XTRAVOL, INCR : REAL);
(*************************************************************)
(*Volume of solution in flask is only shown to increase when a suitable volume (say 5mL
or more) has been released from burette. Therefore smaller increments are summed until
this volume is reached and then level of soln is shown to rise *)
VAR EXTRA : INTEGER;
BEGIN
  XTRAVOL:=XTRAVOL+INCR; (*xtravol. is vol.titrant added that has
                 not yet been shown to fill flask*)
 IF (XTRAVOL>=5.0)THEN (*when xtravol is sufficiently large then flask is filled by
                 an extra amt.  This   value must be even due to slope of flask*)
   BEGIN
     EXTRA:=TRUNC(XTRAVOL/FILLRATE);
     IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
     XTRAVOL:=0;
   END;
END;(*CHECKLEVEL*)


BEGIN   (* TITRATE *)
 INITCONDITIONS(ENDPT1);
 INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
 SELECTINCR(INCR);
 IF NOT QUIT THEN
  BEGIN
   REQUEST;       (*Display prompt to press space bar*)
   SELECTCHANGE:=FALSE;
    REPEAT
      CHECKKEY(SPACEPR,SELECTCHANGE);
      IF SPACEPR THEN
        BEGIN
          MOVEDROP(INCR,OLDLEVEL);
          ADDMORE;
          CHECKINDICATOR;
          CHECKLEVEL(XTRAVOL,INCR);
        END;
      IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
     UNTIL QUIT;
    CHARTYPE(0);
    REQUEST;         (*erase prompt*)
    CHARTYPE(10);
    AGAIN:=(BURVOL>0); (* only give option to repeat 'again' if titration has*)
   END;                                             (*commenced*)
 END; (* TITRATE *)

 (*************************************************************)
 PROCEDURE STARTAGAIN;
 (*************************************************************)
 CONST BLANK='    ';
        X=0;
 VAR CH:CHAR; Y: INTEGER;
      UNSTR: SHORTSTR;
```

```
(*===============================================================*)
PROCEDURE CLEARVALUES;
(*===============================================================*)
(* erase all values plus flask *)

   (*-----------------------------------------------------------*)
   PROCEDURE NEWLABELS(COL : SCREENCOLOR);
   (*-----------------------------------------------------------*)
   CONST  LOWERY=170; TOPY=190;
   VAR SYMBOL : SHORTSTR;

      PROCEDURE LABELS(X :INTEGER; CH1 ,CH2 :SHORTSTR);
      CONST HEIGHT=179;
      BEGIN
       WSTAT(X,HEIGHT,CH1);
       WSTAT(X+7,HEIGHT-7,CH2);
      END;(*LABELS*)

   BEGIN (*NEWLABELS*)
    IF TITRTYPE=ACIDSALT THEN SYMBOL :='b';
    FILLBOX(XMIN,XMAX,LOWERY-2,YMAX,BLACK1);
    DRAWBOX(1,LOWERY,64,TOPY,COL);
    LABELS(7,'M','s');
    DRAWBOX(70,LOWERY,135,TOPY,COL);
    LABELS(78,'M',SYMBOL);
    DRAWBOX(145,LOWERY,206,TOPY,COL);
    LABELS(149,'V','s');
    DRAWBOX(212,LOWERY,275,TOPY,COL);
    LABELS(218,'V',SYMBOL);
   END;(*NEWLABELS*)

BEGIN
 DISPLAYPH(BLANK);
 TITRDISP(BLANK);
 SALTDISP(BLANK);
 FILLBOX(10,110,25,125,BLACK2);     (* erase flask  *)
 FILLBOX(150,271,32,164,BLACK1);    (* erase pH graph *)
 IF NOT AGAIN THEN NEWLABELS(BLUE);
END; (*CLEARVALUES*)

(*===============================================================*)
PROCEDURE CHECKAGAIN;
(*===============================================================*)
CONST DOTS='......(';
BEGIN
 Y:=6;
 WRITE(AT(X,Y),'REPEAT previous titration    ',DOTS,'R)'); Y:=Y+2;
 WRITE(AT(X,Y),'Repeat titration'); Y:=Y+1;
 WRITE(AT(X,Y),'    but ALTER conditions    ',DOTS,'A)'); Y:=Y+2;
 WRITE(AT(X,Y),'Get CALCULATOR              ',DOTS,'C)'); Y:=Y+2;
 WRITE(AT(X,Y),'New assignment              ',DOTS,'N)'); Y:=Y+2;
 WRITE(AT(X,Y),'QUIT - back to MAIN MENU    ',DOTS,'Q)'); Y:=Y+3;
 WRITE(AT(X+10,Y),'SELECT  OPTION  ..',DOTS,'  )');
 GOTOXY(37,Y);
 CLEARVALUES;
 GETTEXTCHAR(X+37,Y,CH,['R','A','C','N','Q']);
```

```
    IF AGAIN THEN AGAIN:=CH='R';
    QUIT:=CH='Q';   (* resets 'quit'  *)
    PAGE(OUTPUT);
    IF NOT AGAIN THEN
     BEGIN
      SALTMOLARITY(BLANK);    (*erase molarity of acid*)
      TITRMOLARITY(BLANK);    (* erase molarity of base*)
     END;
    CASE CH OF
     'C':BEGIN
         GETCALCULATOR;
         CHECKAGAIN;
        END;
     'N':SELECTUNKNOWN;
     END;(*CASE*)
   PAGE(OUTPUT);
  END; (* CHECKAGAIN*)

BEGIN (*STARTAGAIN*)
  PAGE(OUTPUT);
  IF AGAIN THEN CHECKAGAIN ELSE SELECTUNKNOWN;
  CLEARVALUES;
END; (* STARTAGAIN *)


BEGIN   (* main *)
 SETCHAIN(':SALTMENU');
 AGAIN:=FALSE;
 QUIT:=FALSE;
 SETCOLOUR;
 SELECTUNKNOWN;
 INITSCREEN;
 WHILE (NOT QUIT) DO
   BEGIN
    IF NOT AGAIN THEN STANDARDSOLN;
    DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE2);
    GRAFMODE;
    IF ((NOT QUIT) AND (NOT AGAIN)) THEN
       GETCONDITIONS;
    SHOWTYPE;
    IF NOT QUIT THEN TITRATE;
    TEXTMODE;
   STARTAGAIN;
  END;(* while *)
 BACKTOMENU;
END.(*SALTASSIGN*)
```

```
(*$S++*)(*$R-*)(*$V-*)

PROGRAM MIXASSIGN;
USES TURTLEGRAPHICS,TRANSCEND,CHAINSTUFF,USEFUL,SALTLIB;
VAR
   FIRST,NEW,                                  (* flag to select a new assignment *)
   COLOUR,                                          (* is colour monitor available *)
    AGAIN : BOOLEAN;                                (* option to repeat titration  *)
   SALT1CONC,SALT2CONC, TITRCONC: REAL;                          (* conc. of solns  *)
   UNKNOWN: ACIDORBASE;                                 (* type of unknown soln    *)
   STDSOLN :REAL;                              (* concentration of standard soln   *)

(**************************************************************)
PROCEDURE ACIDDISP(S:SHORTSTR);
(**************************************************************)
BEGIN
  WSTAT(7,148,S); (*display acidconc next to flask*)
  WSTAT(20,140,'HCl');
END; (* ACIDDIDSP *)

(**************************************************************)
PROCEDURE ERASEBOXES;
(**************************************************************)
BEGIN
  FILLBOX(5,40,172,188,BLACK2);    (*black2 doesnt interfere *)
  FILLBOX(76,100,172,188,BLACK2);  (*with existing blue boxes*)
  FILLBOX(12,50,172,186,BLACK1);   (*black1 doesnt interfer with*)
  FILLBOX(83,125,172,186,BLACK1);  (*white1 of wstat to follow*)
END; (* ERASEBOXES *)

(**************************************************************)
PROCEDURE NEWLABELS;
(**************************************************************)
VAR X1,X2:INTEGER;

   (*==========================================================*)
   PROCEDURE NEWSTR(X:INTEGER;CH1,CH2:SHORTSTR);
   (*==========================================================*)
   CONST HEIGHT=178;
   BEGIN
    WSTAT(X,HEIGHT,CH1);
    WSTAT(X,HEIGHT-3,CH2);
   END; (* NEWSTR *)

BEGIN (* NEWLABELS *)
 X1:=15; X2:=85;
 CHARTYPE(6);
 NEWSTR(X1,'Na CO',' 2 3');
 NEWSTR(X2,'NaHCO','    3');
 CHARTYPE(10);
END; (* NEWLABELS *)
```

```
(**************************************************************)
PROCEDURE GETCALCULATOR;
(**************************************************************)
CONST  TOP=10;  INDENT=20;
VAR Y: INTEGER;
     NUM1,NUM2: REAL;
     ESCAPE:BOOLEAN;
     OP:CHAR;

   (*==========================================================*)
   PROCEDURE LAYOUT;
   (*==========================================================*)
   CONST STAR='*';
   VAR X,Y:INTEGER;
   BEGIN
     PAGE(OUTPUT);
     X:=10; Y:=2;
     WRITE(AT(X,Y),AROW(21,STAR));     Y:=Y+2;
     WRITE(AT(X,Y),' C A L C U L A T O R'); Y:=Y+2;
     WRITE(AT(X,Y),AROW(21,STAR));
     WRITE(AT(X,23),'<Q> QUIT     <C> CLEAR');
     X:=4; Y:=TOP;
     WRITE(AT(X,Y),'+');Y:=Y+2;
     WRITE(AT(X,Y),'-');Y:=Y+2;
     WRITE(AT(X,Y),'X');Y:=Y+2;
     WRITE(AT(X,Y),'/');
   END; (* LAYOUT *)

   (*==========================================================*)
   PROCEDURE ENTERNUM(Y:INTEGER; VAR NUM:REAL);
   (*==========================================================*)
   CONST X=10;
   VAR S:SHORTSTR;
   BEGIN
     WRITE(AT(X,Y),'ENTER NUM:');
     GETRESPONSE(22,Y,S,8,['0'..'9','.','Q','C']);
     WRITE(AT(X,Y),'          ');
     QUIT:=((S='Q') OR (S='C'));
     ESCAPE:=S='Q';
     NUM:=RVALUE(S);
   END; (* ENTERNUM *)

   (*==========================================================*)
   PROCEDURE ENTEROP(Y:INTEGER; VAR OP:CHAR);
   (*==========================================================*)
   BEGIN
     GOTOXY(INDENT,Y);
     GETACHAR(OP,['+','-','X','*','/','Q','C']);
     WRITE(OP);
     QUIT:=((OP='Q') OR (OP='C'));
     ESCAPE:=OP='Q';
   END; (* ENTEROP *)
```

```
(*==================================================================*)
PROCEDURE CLEAR(A,B:INTEGER);
(*==================================================================*)
CONST BLANKL='         ';
VAR   J: INTEGER;
BEGIN
   FOR J:=A TO B DO WRITE(AT(INDENT,J),BLANKL);
END; (* CLEAR *)

(*==================================================================*)
PROCEDURE CALC(VAR NUM1,NUM2:REAL; OP:CHAR);
(*==================================================================*)
BEGIN
 CASE OP OF
     '+': NUM1:=NUM1+NUM2;
     '-': NUM1:=NUM1-NUM2;
 'X','*': NUM1:=NUM1*NUM2;
     '/': NUM1:=NUM1/NUM2;
   END; (*CASE*)
  WRITE(AT(INDENT,Y),NUM1:9:5);
 END; (* CALC *)

BEGIN (* CALCULATOR *)
  LAYOUT;
  REPEAT
     Y:=TOP;
     ENTERNUM(Y,NUM1);
     IF NOT QUIT THEN
      REPEAT
          Y:=Y+2;
          ENTEROP(Y,OP);
          IF NOT QUIT THEN
           BEGIN
            Y:=Y+2;
            ENTERNUM(Y,NUM2);
            CLEAR(TOP,Y);
            Y:=TOP;
           END;
          IF NOT QUIT THEN CALC(NUM1,NUM2,OP);
        UNTIL QUIT;
     CLEAR(TOP,Y);
  UNTIL ESCAPE;
  QUIT:=FALSE;
  PAGE(OUTPUT);
END; (* CALCULATOR *)

(******************************************************************)
PROCEDURE GETSPACEBAR;
(******************************************************************)
VAR CH:CHAR;
BEGIN
  WRITE(AT(23,23),'Press <SPACE BAR>');
  GETACHAR(CH,[SPACE,'Q']);
  QUIT:=CH='Q';
END; (* GETSPACEBAR *)
```

```
(************************************************************)
PROCEDURE SELECTUNKNOWN;
(************************************************************)
CONST  STAR='*';
VAR  UNKNOWN1,UNKNOWN2:REAL;
      X,Y:INTEGER;
      ASSIGN:STRING[2];
      CH:CHAR;
BEGIN (* SELECTUNKNOWN *)
   PAGE(OUTPUT);
   X:=0; Y:=6;
   WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
   WRITE(AT(X,Y),'ENTER ASSIGNMENT NO.(1-99):'); Y:=Y+2;
   WRITE(AT(X,Y),'This no. will be given to you by'); Y:=Y+1;
   WRITE(AT(X,Y),'your teacher.'); Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,STAR));
   GETRESPONSE(X+35,Y-5,ASSIGN,2,NUMS+['Q']);
   QUIT:=(POS('Q',ASSIGN)>0);
   IF QUIT THEN EXIT(SELECTUNKNOWN);
   IF LENGTH(ASSIGN)<2 THEN  ASSIGN:=CONCAT('0',ASSIGN);
   TITRTYPE:=DISALT;
   UNKNOWN:=ASALT;

   UNKNOWN1:=ORD(ASSIGN[1])-48;
   UNKNOWN2:=ORD(ASSIGN[2])-48;
   SALT1CONC:=(UNKNOWN2*8+UNKNOWN1)/100;
   SALT2CONC:=(UNKNOWN1*5+UNKNOWN2)/100;
   PAGE(OUTPUT);
   X:=0;Y:=2;
   WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
   WRITE(AT(X+3,Y),'YOUR ASSIGNMENT IS TO CALCULATE'); Y:=Y+2;
   WRITE(AT(X,Y),'THE CONCENTRATIONS OF SODIUM CARBONATE'); Y:=Y+2;
   WRITE(AT(X,Y),'AND SODIUM BICARBONATE IN A SOLUTION.');  Y:=Y+3;
   WRITE(AT(X+3,Y),'CONCENTRATIONS WILL BE IN RANGE');Y:=Y+2;
   WRITE(AT(X+10,Y),'0.010-1.000M');Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,STAR));
   WRITE(AT(10,22),'Press <SPACE BAR> to continue ');
   GETACHAR(CH,[SPACE,'Q']);
   QUIT:=CH='Q';
   PAGE(OUTPUT);
END; (* SELECTUNKNOWN *)

(************************************************************)
PROCEDURE STANDARDSOLN;
(************************************************************)
CONST  STAR='*';
       X=0;  BLANK='        ';
VAR  S:STRING; CH:CHAR;
     OK:BOOLEAN; Y : INTEGER;
BEGIN (* STANDARDSOLN *)
  Y:=8;
  WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
  WRITE(AT(X,Y),'Enter concentration of standard'); Y:=Y+2;
  WRITE(AT(X,Y),'solution of hydrochloric acid.');  Y:=Y+2;
  WRITE(AT(X,Y),'(0.010-1.000): ');
  WRITE(AT(X,Y+2),AROW(40,STAR));
```

```
  S:='';
  REPEAT
     GETRESPONSE(20,Y,S,5,NUMS+['Q']);
     QUIT:=S='Q';
     IF NOT QUIT THEN
       BEGIN
         STDSOLN:=RVALUE(S);
         OK:=((STDSOLN>=0.01) AND (STDSOLN<=1.0));
         IF NOT OK THEN WRITE(AT(20,Y),BLANK);
       END;
  UNTIL OK OR QUIT;
  IF NOT QUIT THEN TITRCONC:=STDSOLN;
  PAGE(OUTPUT);
 END; (* STANDARDSOLN *)

(*****************************************************************)
PROCEDURE GETCONDITIONS;
(*****************************************************************)
 VAR ASTR: SHORTSTR;
     PROMPT1,PROMPT2:STRING;

   (*=============================================================*)
   PROCEDURE SELECTVOL(VAR VOL: REAL);
   (*=============================================================*)
   CONST  MIN=10.0; MAX=50;      (* range of volume*)
          X=215; Y=10;          (* coord. to enter input *)
   BEGIN
      PROMPT1:=CONCAT('Select vol. of salt in flask:');
      PROMPT2:=' (10.0-50.0mL)';
      TWOPROMPTS(PROMPT1,PROMPT2);
      INRANGERESPONSE(VOL,ASTR,MIN,MAX,X,Y);
      CHARTYPE(6);
      TWOPROMPTS(PROMPT1,PROMPT2); (* ERASE *)
      CHARTYPE(10);
   END; (* SELECTVOL *)

   (*=============================================================*)
   PROCEDURE SELECT(VAR INFLASK:ACIDORBASE);
   (*=============================================================*)
   VAR CH:CHAR;
   BEGIN
      INFLASK:=ASALT;
      PROMPT1:='Salts solution is in flask';
      PROMPT2:='        Press <SPACE BAR> to continue';
      TWOPROMPTS(PROMPT1,PROMPT2);
      GETACHAR(CH,[SPACE,'Q']);
      QUIT:=CH='Q';
      CHARTYPE(6);
      TWOPROMPS(PROMPT1,PROMPT2);  (* ERASE *)
      CHARTYPE(10);
   END; (* SELECT *)

BEGIN (* GETCONDITIONS *)
 K1:=4.3E-7;
 K2:=5.6E-11;(* K values for carbonic *)
 REALSTR(TITRCONC,ASTR,3,5);
```

```
 ACIDDISP(ASTR);
 IF NOT QUIT THEN SELECT(INFLASK);
 IF NOT QUIT THEN SELECTVOL(FLASKVOL);
END; (* GETCONDITIONS *)

(*****************************************************************)
PROCEDURE SHOWTYPE;
(*****************************************************************)
VAR X,Y,K: INTEGER;
     S: ARRAY [1..3] OF STRING[18];
BEGIN
   S[1]:='MIXTURE OF';
   S[2]:='SODIUM CARBONATE &';
   S[3]:='SODIUM BICARBONATE';
   X:=210; Y:=135;
   FOR K:=1 TO 3 DO
    BEGIN
     WSTAT(X-(7*LENGTH(S[K]) DIV 2),Y,S[K]);
     Y:=Y-20;
    END;
   Y:=Y-30;
   WSTAT(X-(7*8),Y,'ACID IN BURETTE');
END; (* SHOWTYPE *)

(*$I :MIXASS2*)
(*$I :MIXASS3*)
```

```
(*MIXASS2 - INCLUDED IN MIXASSIGN*)
(*****************************************************************)
PROCEDURE CALCULPH(ANYSALT1,ANYSALT2,SALTVOL,ACIDVOL,ACIDCONC:REAL;
          VAR PH: REAL);
(*****************************************************************)
(*ANYSALT1 & 2 - initial moles of both salts *)
VAR
 WACID,SACID,ACIDMOL, TOTALVOL,
 SALT1CONC, SALT2CONC, EXCESS  : REAL;
 A,B,C,D,APPROX :REAL;
 FIRST : BOOLEAN;

   (*=============================================================*)
   PROCEDURE NEWTONCUBIC(VAR PH:REAL);
   (*=============================================================*)
   (*iterates cubic equation - requires global APPROX and constants A,B,C,D*)
   VAR
    COUNT:INTEGER;
    NEWTONX,ERROR,ONEPER, GUESS:  REAL;
    SOLN: BOOLEAN;

      (*----------------------------------------------------------*)
      FUNCTION EQUATION(X:REAL):REAL;
      (*----------------------------------------------------------*)
      BEGIN
       EQUATION:=D+X*(C+X*(B+A*X));
      END; (*EQUATION*)

      (*----------------------------------------------------------*)
      FUNCTION DERIV(X:REAL):REAL;
      (*----------------------------------------------------------*)
      BEGIN
       DERIV:=C+X*(3*A*X+2*B);
      END; (*DERIV*)

   BEGIN (* NEWTONCUBIC *)
    COUNT:=0;
    GUESS:=APPROX;
    SOLN:=FALSE;
    REPEAT
       COUNT:=COUNT+1;
       NEWTONX:=APPROX-(EQUATION(APPROX)/DERIV(APPROX));
       ERROR:=ABS(APPROX-NEWTONX);
       ONEPER:=NEWTONX * 0.01;
       IF (ERROR<ONEPER) THEN  SOLN:=TRUE  ELSEAPPROX:=NEWTONX;
     UNTIL ((COUNT>20) OR (SOLN));
    IF (NEWTONX<0.0) THEN
     BEGIN (* If negative root, make another guess & iterate until +ve soln *)
      APPROX:=GUESS*10;
      NEWTONCUBIC(PH);
     END
      ELSE PH:=-1*LOG(NEWTONX);
   END; (* NEWTONCUBIC *)
```

```
(*=====================================================================*)
PROCEDURE QUADRATIC(WACID :REAL; VAR H,PH :REAL);
(*=====================================================================*)
(*Soln of salt of weak acid/strong base titrated with s.acid. H+ results from excess of
 strong & hydrolysis of salt is often insignificant *)
VAR HYDROL : REAL;
BEGIN
   HYDROL :=SQRT(K1 *WACID); (*H+ from hydrolysis of weak acid*)
   H :=H + HYDROL;
   PH :=-1 *LOG(H);
END; (* QUADRATIC *)

(*=====================================================================*)
PROCEDURE APPROX1(ACIDCONC :REAL; VAR PH :REAL);
(*=====================================================================*)
BEGIN
   A :=1 /K1 ;
   B :=1 ;
   C :=K2-ACIDCONC;
   D :=-2.0*K2*ACIDCONC;
   IF K1 >=1.0 THEN APPROX :=ACIDCONC
     ELSE APPROX :=SQRT(K1 *ACIDCONC);
   NEWTONCUBIC(PH);
END; (* APPROXCUB1 *)

(*=====================================================================*)
PROCEDURE APPROX2(ACIDCONC,SALTCONC :REAL; VAR PH :REAL);
(*=====================================================================*)
BEGIN
  APPROX :=K1 *ACIDCONC/SALTCONC;
  IF K1 >=1.0 THEN APPROX :=0.01 *APPROX;
  IF APPROX>=10E-7 THEN
   BEGIN
    A :=1 /K1 ;
    B :=(SALTCONC/K1)+1 ;
    C :=K2-ACIDCONC;
    D :=-1.0*K2*(SALTCONC+2*ACIDCONC);
   END

   ELSE
    BEGIN
      A :=SALTCONC/(K1 *K2);
      B :=(KW/K1)+ACIDCONC;
      B :=-1 *B/K2;
      C :=-1 *(SALTCONC+(2*ACIDCONC)+KW/K2);
      D :=-KW;
     END;
  NEWTONCUBIC(PH);
 END; (*APPROX2*)

(*=====================================================================*)
PROCEDURE APPROX3(SALTCONC :REAL; VAR PH :REAL);
(*=====================================================================*)
BEGIN
  APPROX :=SQRT(K1 *K2);
  IF APPROX>=1.0E-7 THEN
```

```
  BEGIN
   A:=1/K1;
   B:=(SALTCONC/K1) + 1;
   C:=K2;
   D:=-1.0*(K2*SALTCONC);
  END

  ELSE
   BEGIN
    A:=SALTCONC/(KW*K1);
    B:=-1.0/K1;
    C:=(K2*SALTCONC/KW)+1;
    C:=-1.0*C;
    D:=-1.0*K2;
   END;
 NEWTONCUBIC(PH);
END; (* APPROXCUB3 *)

(*============================================================*)
PROCEDURE APPROX4(SALT1,SALT2:REAL; VAR PH:REAL);
(*============================================================*)
BEGIN
 APPROX:=K2*SALT1/SALT2;
 IF APPROX>=1.0E-7 THEN
  BEGIN
   A:=1/K1;
   B:=((SALT1+2*SALT2)/K1)+1;
   C:=K2+SALT2;
   D:=-1.0*(K2*SALT1);
  END

  ELSE
  BEGIN
   A:=(SALT1+2*SALT2)/K1;
   B:=SALT2-(KW/K1);
   C:=(K2*SALT1)+KW;
   C:=-1.0*C;
   D:=-1.0*KW*K2;
  END;
 NEWTONCUBIC(PH);
END; (* APPROXCUB4 *)

(*==================================================================*)
PROCEDURE INITCALC;
(*==================================================================*)
(* passes volume of acid & base to this procedure but returns  moles of acid & base
 also returns concentration of species in excess if acid>base then excess will be a +ve
 value otherwise it will be -ve     *)
CONST DIFF=0.000001;
BEGIN
 TOTALVOL:=SALTVOL+ACIDVOL;
 ACIDMOL:=ACIDVOL*ACIDCONC;
 SALT1CONC:=ANYSALT1/TOTALVOL;
 SALT2CONC:=ANYSALT2/TOTALVOL;
 EXCESS:=ANYSALT1-ACIDMOL;
 IF ABS(EXCESS)<DIFF THEN EXCESS:=0.0;
```

```
    FIRST:=EXCESS>=0.0;
    IF NOT FIRST THEN
      BEGIN   (* past 1st end pt*)
        EXCESS:=(ANYSALT2 + 2*ANYSALT1)-ACIDMOL;
        IF ABS(EXCESS)<DIFF THEN EXCESS:=0.0;
      END;
  END; (* INITCALC *)


BEGIN  (* CALCPH *)
  INITCALC;   (* vol. of anyacid & anybase converted into moles *)
  IF FIRST THEN        (*salt1>titrant ie. before lst end pt *)
   BEGIN              (* or start of titration. acid=0      *)
    IF EXCESS=0.0 THEN  (* or 1st end pt *)
      BEGIN
       SALT2CONC:=(ACIDMOL+ANYSALT2)/TOTALVOL;
       APPROX3(SALT2CONC,PH);
      END
      ELSE
      BEGIN
       SALT1CONC:=EXCESS/TOTALVOL;
       SALT2CONC:=(anysalt2 + acidmol)/totalvol;
       APPROX4(SALT2CONC,SALT1CONC,PH);
      END;
   END

   ELSE
   BEGIN
    IF EXCESS>0.0 THEN
     BEGIN
      SALT2CONC:=EXCESS/TOTALVOL; (*between 1st - 2nd endpt *)
      WACID:=(ACIDMOL-ANYSALT1)/TOTALVOL ;
      APPROX2(WACID,SALT2CONC,PH);
     END

    ELSE
     IF EXCESS=0.0 THEN        (* 2nd end pt *)
     BEGIN
      WACID:=(ANYSALT1+ANYSALT2)/TOTALVOL;
      APPROX1(WACID,PH);
     END

     ELSE                 (*Past 2nd end pt *)
      BEGIN
       WACID:=ANYSALT1+ANYSALT2;
       SACID:=ACIDMOL-(2*ANYSALT1+ANYSALT2);
       WACID:=WACID/TOTALVOL;
       SACID:=SACID/TOTALVOL;
       QUADRATIC2(WACID,SACID,PH);
      END;
   END; (*ELSE*)

END; (* CALCULPH *)
```

```
(*MIXASS3 - included in MIXASSIGN *)
(**************************************************************)
PROCEDURE WHICHIND;
(**************************************************************)
CONST STAR='*';
VAR X,Y:INTEGER;
    CH:CHAR;
BEGIN
 PAGE(OUTPUT);
 X:=0; Y:=2;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+2;
 WRITE(AT(X+6,Y),'INDICATOR COLOUR CHANGE');Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,STAR)); Y:=Y+3;
 WRITE(AT(X,Y),'Hypothetical indicator will change'); Y:=Y+2;
 WRITE(AT(X,Y),'colour at'); Y:=Y+3;
 WRITE(AT(X,Y),'1st equivalence pt         ....(1)'); Y:=Y+2;
 WRITE(AT(X,Y),'2nd equivalence pt         ....(2)'); Y:=Y+3;
 WRITE(AT(X+8,Y),'SELECT OPTION ...... (  )');
 GETTEXTCH(X+37,Y,CH,['1','2','Q']);
 QUIT:=CH='Q';
 FIRST:=CH='1';
END; (* WHICHENDPT *)

(**************************************************************)
PROCEDURE TITRATE;
(**************************************************************)
VAR
   SALTCOL,TITRCOL,                          (* soln colours during titration    *)
   SOLNCOL : SCREENCOLOR;                    (* current colour of soln in flask  *)
   SPACEPR,                                  (* flag to indicate space bar pressed *)
   SELECTCHANGE: boolean;       (* flag to indicate change in titrant increment volume *)
   INCR,                                     (* current titrant increment vol.   *)
   BURVOL,                          (* total vol. added from burette (titrant)  *)
   SALTVOL,TITRVOL,                     (* total vol. of salt & titrant in flask *)
   ENDPT,
   SMOL1,SMOL2,                              (* initial moles of both salts  *)
   PH,                                       (* current pH of soln   *)
   XTRAVOL                           (* vol. titrant not yet shown to fill flask *)
   PHRATIO : REAL;    (*ratio between no. pixels on pHscale & pH range to be plotted *)
   RTSIDE,LTSIDE,                       (* current x-coord. of flask being filled  *)
   OLDLEVEL,INCREASE,                   (* current & increase in level of soln   *)
   OLDX,OLDY:INTEGER;     (*current coord of pH graph required for graphing pH curve*)

   (*=========================================================*)
   PROCEDURE INITCONDITIONS;
   (*=========================================================*)
   VAR  VOLSTR: SHORTSTR;
        ENDPT1,ENDPT2:REAL;
   BEGIN
    IF COLOUR THEN
     BEGIN SALTCOL:=VIOLET; TITRCOL:=BLUE; END
      ELSE
        BEGIN SALTCOL:=WHITE1; TITRCOL:=BLACK1; END;
    SOLNCOL:=SALTCOL;
    BURVOL:=0.0;
    SALTVOL:=FLASKVOL;
```

```
 TITRVOL:=BURVOL;
 SMOL1:=SALTVOL*SALT1CONC;
 SMOL2:=SALTVOL*SALT2CONC;
 ENDPT1:=SMOL1/TITRCONC;
 ENDPT2:=SMOL2/TITRCONC;
 ENDPT2:=ENDPT2 + 2*ENDPT1;
 IF FIRST THEN ENDPT:=ENDPT1 ELSE ENDPT:=ENDPT2;
  FILLRATE:=2;  (*determines rate at which flask filled*)

  REALSTR(SALTVOL,VOLSTR,2,6);
  SALTDISP(VOLSTR);           (* Display SALT volume *)
  REALSTR(TITRVOL,VOLSTR,2,6);
  TITRDISP(VOLSTR);          (* Display ACID volume *)
  IF SMOL2=0 THEN PH:=-LOG(SQRT(KW*K2/SALT1CONC))   (*only carbonate*)
    ELSE
      CALCULPH(SMOL1,SMOL2,SALTVOL,TITRVOL,TITRCONC,PH);
  REALSTR(PH,VOLSTR,2,5);
  DISPLAYPH(VOLSTR);
 END;  (*INITCONDITIONS*)

(*================================================================*)
PROCEDURE INITLEVEL(VAR RIGHTS,LEFTS,TOPLEVEL:INTEGER);
(*================================================================*)
(* initializes coord of sides of flask & top of flask as well  as level of soln in flask   *)
CONST WIDTH=2; (* indent soln from sides of flask *)
BEGIN
 FLASKTOP:=FLASKY+ (3*FLASKSIZ)DIV 4; (*y-coord. of top sloping sides of flask*)
 NECKTOP:=FLASKY+FLASKSIZE; (*y-coord. of very top of flask*)
 LEFTS:=FLASKX-(FLASKSIZ DIV 2)+WIDTH;   (*calc. coord of *)
 RIGHTS:=FLASKX+(FLASKSIZ DIV 2)-WIDTH;  (*sides of flask given midpt.of base*)
 TOPLEVEL:=FLASKY+1;           (* base of flask= Flasky*)
 INCREASE:=10;   (*depth of soln initially placed in flask*)
 XTRAVOL:=0.0;          (*initialize increment in titrant *)
END;  (* INITLEVEL *)

(*================================================================*)
PROCEDURE ADDMORE;
(*================================================================*)
(* Increment vol of titrant & calc new pH; display new pH & new volume of titrant*)
CONST BLANK='      ';
VAR  VOL: INTEGER; PHSTR,VOLSTR:SHORTSTR;
BEGIN  (* ADDMORE*)
 BURVOL:=BURVOL+INCR; (* calculate total vol. of titrant *)
 VOL:=ROUND(BURVOL*100);    (* this prevents build up of *)
 BURVOL:=VOL/100.0;         (* floating point errors *)
 REALSTR(BURVOL,VOLSTR,2,6); (* convert vol.  to string   *)

 TITRVOL:=BURVOL;    (* salt must be in flask*)
 TITRDISP(BLANK);
 TITRDISP(VOLSTR);

 CALCULPH(SMOL1,SMOL2,SALTVOL,TITRVOL,TITRCONC,PH);
 DISPLAYPH(BLANK);            (* Erase old pH      *)
 REALSTR(PH,PHSTR,2,5);          (* convert to string *)
 DISPLAYPH(PHSTR);            (* Display new pH    *)
END;   (* ADDMORE *)
```

```
(*=================================================================*)
PROCEDURE CHANGECOL(NEWCOLOR:SCREENCOLOR);
(*=================================================================*)
(*Change colour of soln & change label of soln in flask*)
VAR CURRENTL,DEPTH : INTEGER;
BEGIN
 SOLNCOL:=NEWCOLOR;
 CURRENTL:=OLDLEVEL;
 INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
 DEPTH:=CURRENTL-OLDLEVEL;
 FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
 IF CURRENTL>OLDLEVEL THEN     (* colour change with soln in neck of flask*)
   BEGIN
     DEPTH:=CURRENTL-OLDLEVEL;
     FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,DEPTH,NEWCOLOR);
   END;
END; (*CHANGECOL*)

(*=================================================================*)
PROCEDURE CHECKINDICATOR;
(*=================================================================*)
BEGIN
  IF ((SOLNCOL=SALTCOL) AND (BURVOL>=ENDPT))
    THEN CHANGECOL(TITRCOL);
END; (* CHECKINDICATOR *)

(*=================================================================*)
PROCEDURE CHECKLEVEL(VAR XTRAVOL,INCR:REAL);
(*=================================================================*)
(*Volume of solution in flask is only shown to increase when a suitable volume (say 5mL
 or more) has been released from  burette. Therefore smaller increments are summed
 until this  volume is reached and then level of soln is shown to rise *)
VAR EXTRA : INTEGER;
BEGIN
  XTRAVOL:=XTRAVOL+INCR; (*xtravol. is vol.titrant added that  has not yet been
                                                        shown to fill flask*)
 IF (XTRAVOL>=5.0)THEN  (*when xtravol is sufficiently large then flask is filled by
                         an extra amt. This value must be even due to slope of flask*)
   BEGIN
      EXTRA:=TRUNC(XTRAVOL/FILLRATE);
      IF ODD(EXTRA) THEN EXTRA:=EXTRA-1;
      FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,EXTRA,SOLNCOL);
      XTRAVOL:=0;
   END;
END; (* CHECKLEVEL *)

BEGIN   (* TITRATE *)
  INITCONDITIONS;
  INITLEVEL(RTSIDE,LTSIDE,OLDLEVEL);
  FILLFLASK(LTSIDE,RTSIDE,OLDLEVEL,INCREASE,SOLNCOL);
  AGAIN:=TRUE;
  SELECTINCR(INCR);
  IF NOT QUIT THEN
   BEGIN
     REQUEST;       (*Display prompt to press space bar*)
     SELECTCHANGE:=FALSE;
```

```
      REPEAT
        CHECKKEY(SPACEPR,SELECTCHANGE);
        IF SPACEPR THEN
          BEGIN
            MOVEDROP(INCR,OLDLEVEL);
            ADDMORE;
            CHECKINDICATOR;
            CHECKLEVEL(XTRAVOL,INCR);
          END;
        IF SELECTCHANGE THEN CHANGEINC(SELECTCHANGE,INCR);
      UNTIL QUIT;
     CHARTYPE(0);
     REQUEST;          (*erase prompt*)
     CHARTYPE(10);
     AGAIN:=(BURVOL>0); (* only give option to repeat  if  titration has commenced*)
   END;
END; (* TITRATE *)

(***********************************************************)
PROCEDURE STARTAGAIN;
(***********************************************************)
CONST  BLANK='    ';
   (*=======================================================*)
   PROCEDURE CLEARVALUES;
   (*=======================================================*)
   BEGIN                                       (* erase all values plus flask *)
     DISPLAYPH(BLANK);     (* delete pH *)
     TITRDISP(BLANK);      (* delete vol. of SALT*)
     IF NOT AGAIN THEN
      BEGIN
       SALTDISP(BLANK);       (* delete vol. of salt*)
       ACIDDISP(BLANK);      (* delete molarity of acid*)
      END;
     FILLBOX(10,110,25,125,BLACK1);  (* erase flask  *)
   END; (*CLEARVALUES*)

   (*=======================================================*)
   PROCEDURE CHECKAGAIN;
   (*=======================================================*)
   CONST DOTS='......(';
   VAR X,Y : INTEGER;  CH:CHAR;
   BEGIN
    Y:=6; X:=0;
    WRITE(AT(X,Y),'REPEAT previous titration    ',DOTS,'R)'); Y:=Y+2;
    WRITE(AT(X,Y),'Get CALCULATOR               ',DOTS,'C)');  Y:=Y+2;
    WRITE(AT(X,Y),'New assignment               ',DOTS,'N)');  Y:=Y+2;
    WRITE(AT(X,Y),'QUIT - back to MAIN MENU     ',DOTS,'Q)');  Y:=Y+3;
    WRITE(AT(X+10,Y),'SELECT  OPTION  ..',DOTS,'  )');
    GOTOXY(37,Y);
    GETTEXTCHAR(X+37,Y,CH,['R','C','N','Q']);
    QUIT:=CH='Q';  (* resets 'quit'  *)
    PAGE(OUTPUT);
    CASE CH OF
     'C' :BEGIN
         GETCALCULATOR;
         CHECKAGAIN;
```

```
          END;
        'N':BEGIN
            NEW:=TRUE;
            AGAIN:=FALSE;
          END;
        'R':IF AGAIN THEN
          BEGIN
            Y:=5;
            WRITE(AT(X,Y),'Do you wish to alter either:'); Y:=Y+4;
            WRITE(AT(X+6,Y),'(i) acid concentration'); Y:=Y+2;
            WRITE(AT(X+10,Y),'OR'); Y:=Y+2;
            WRITE(AT(X+5,Y),'(ii) volume of solution in flask'); Y:=Y+5;
            WRITE(AT(X+6,Y),'(Y/N)');
            GETTEXTCHAR(X+12,Y,CH,['Y','N']);
            AGAIN:=CH='N';
          END;
        END; (*CASE*)
      CLEARVALUES;
      PAGE(OUTPUT);
    END; (* CHECKAGAIN *)
  BEGIN (* STARTAGAIN *)
    PAGE(OUTPUT);
    NEW:=FALSE;
    CHECKAGAIN;
END; (* STARTAGAIN *)

BEGIN   (* main *)
    SETCHAIN(':SALTMENU');
    AGAIN:=FALSE;
    QUIT:=FALSE;
    NEW:=TRUE;
    INITSCREEN;
    ERASEBOXES;
    NEWLABELS;
    SETCOLOUR;
    WHILE (NOT QUIT) DO
      BEGIN
        IF NEW THEN SELECTUNKNOWN;
        IF NOT QUIT THEN
          BEGIN
            IF NOT AGAIN THEN STANDARDSOLN;
            DRAWFLASK(FLASKX,FLASKY,FLASKSIZ,WHITE1);
            IF NOT QUIT THEN WHICHIND;
            GRAFMODE;
            IF (NOT AGAIN) AND (NOT QUIT) THEN GETCONDITIONS;
            SHOWTYPE;
            IF NOT QUIT THEN TITRATE;
            TEXTMODE;
          END;
        STARTAGAIN;
      END; (* while *)
    BACKTOMENU;
END. (*MIXASSIGN*)
```

```
(*$S++*)
(* This is SYSTEM.STARTUP for macroscopic/microscopic demonstrations *)

PROGRAM INTRO;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;

CONST MODE=6;
VAR  QUIT:BOOLEAN;

(***************************************************************)
PROCEDURE ENCLOSE(COL:SCREENCOLOR);
(***************************************************************)
VAR X1,X2,Y1,Y2,WIDTH,CENTRE :INTEGER;
BEGIN
 X1:=XMIN;X2:=XMAX;
 Y1:=YMIN;Y2:=YMAX;
 CENTRE:=(YMAX DIV 2)-1;
 WIDTH:=12;
 MOVECOL(X1,Y1,COL);
 REPEAT
   MOVETO(X1,Y2);
   MOVETO(X2,Y2);
   MOVETO(X2,Y1);
   MOVETO(X1,Y1);
   MOVETO(X1,Y2);
   X1:=X1+1; X2:=X2-1;
   Y1:=Y1+1; Y2:=Y2-1;
 UNTIL Y1>CENTRE;
 PENCOLOR(BLACK1);
 REPEAT
   MOVETO(X2,Y2);
   MOVETO(X1,Y2);
   MOVETO(X1,Y1);
   MOVETO(X2,Y1);
   MOVETO(X2,Y2);
   X1:=X1-1; X2:=X2+1;
   Y1:=Y1-1; Y2:=Y2+1;
 UNTIL X1<WIDTH;
 PENCOLOR(NONE);
END; (* ENCLOSE *)

(***************************************************************)
PROCEDURE TITLE;
(***************************************************************)
CONST
   TTUBEX=60;
   TTUBEY=36;
   TTWIDTH=32;
   TTSIZE=116;
   TTLEVEL=115;
  TYPE
   BUBSHAPE=PACKED ARRAY[0..5,0..7] OF BOOLEAN;
   BITSHAPE=PACKED ARRAY[0..5,0..15] OF BOOLEAN;
```

```
VAR  METAL:BITSHAPE;
     BUBBLE:BUBSHAPE;
     SPEED,COUNT,BUBBLX,BUBBLY,
     DY,SKIP,GAP:INTEGER;

 (*==============================================================*)
 PROCEDURE BORDER(COL:SCREENCOLOR);
 (*==============================================================*)
 VAR X1,X2,Y1,Y2,WIDTH :INTEGER;
 BEGIN
  X1:=XMIN;X2:=XMAX;
  Y1:=YMIN;Y2:=YMAX;
  WIDTH:=12;
  MOVECOL(X1,Y1,COL);
  REPEAT
   MOVETO(X1,Y2);
   MOVETO(X2,Y2);
   MOVETO(X2,Y1);
   MOVETO(X1,Y1);
   MOVETO(X1,Y2);
   X1:=X1+1; X2:=X2-1;
   Y1:=Y1+1; Y2:=Y2-1;
  UNTIL X1>WIDTH;
  PENCOLOR(NONE);
 END; (* BORDER *)

 (*==============================================================*)
 PROCEDURE PAINT(X,Y,TOP:INTEGER; COL:SCREENCOLOR);
 (*==============================================================*)

   (*------------------------------------------------------------*)
   FUNCTION LOWEST(VAR Y:INTEGER):INTEGER;
   (*------------------------------------------------------------*)
   BEGIN
    WHILE (NOT SCREENBIT(X,Y)) AND (NOT SCREENBIT(X+1,Y))AND (Y>0)
       DO Y:=Y-1;
    Y:=Y+1;
    LOWEST:=Y;
   END; (* LOWEST *)

   (*------------------------------------------------------------*)
   FUNCTION RIGHTMOST(X:INTEGER):INTEGER;
   (*------------------------------------------------------------*)
   BEGIN
    WHILE (NOT SCREENBIT(X,Y)) AND (X<XMAX) DO X:=X+1;
    RIGHTMOST:=X-1;
   END; (* RIGHTMOST *)

   (*------------------------------------------------------------*)
   FUNCTION LEFTMOST(X:INTEGER):INTEGER;
   (*------------------------------------------------------------*)
   BEGIN
    WHILE (NOT SCREENBIT(X,Y)) AND (X>0) DO X:=X-1;
    LEFTMOST:=X+1;
   END; (* LEFTMOST *)
```

```
BEGIN (* PAINT *)
 MOVETO(X,LOWEST(Y));
 WHILE (NOT SCREENBIT(X,Y)) AND (NOT SCREENBIT(X+1,Y)) AND (Y<TOP) DO
   BEGIN
     MOVETO(X,Y);
     MOVETO(RIGHTMOST(X),Y);
     PENCOLOR(COL);
     MOVETO(LEFTMOST(X),Y);
     PENCOLOR(NONE);
     Y:=Y+1;
   END;
END; (* PAINT *)

(*=================================================================*)
PROCEDURE DRAWTTUBE(TUBEX,TUBEY,WIDTH,SIZE,LEVEL:INTEGER;
                                                   COL:SCREENCOLOR);
(*=================================================================*)
VAR EIGHTH,SIXNTH,RWIDTH:REAL;
     X,Y:ARRAY[1..15]OF INTEGER;
     J: INTEGER;
BEGIN
 RWIDTH:=WIDTH;
 EIGHTH:=RWIDTH/8;
 Y[8]:=ROUND(EIGHTH*1.5);
 Y[4]:=ROUND(EIGHTH*1.25);
 Y[12]:=Y[4];
 Y[2]:=ROUND(EIGHTH*0.75);
 Y[14]:=Y[2];
 Y[1]:=ROUND(EIGHTH*0.5);
 Y[15]:=Y[1];
 Y[3]:=(Y[2]+Y[4])DIV 2;
 Y[13]:=Y[3];
 Y[6]:=ROUND(EIGHTH*1.4);
 Y[10]:=Y[6];
 Y[5]:=ROUND(EIGHTH*1.32);
 Y[11]:=Y[5];
 Y[7]:=Y[8];
 Y[9]:=Y[8];

 SIXNTH:=RWIDTH/16;
 FOR J:=1 TO 15 DO X[J]:=ROUND(SIXNTH*J);
 MOVECOL(TUBEX,TUBEY+SIZE,COL);
 MOVETO(TUBEX,TUBEY);
 FOR J:=1 TO 15 DO MOVETO(TUBEX+X[J],TUBEY-Y[J]);
 MOVETO(TUBEX+WIDTH,TUBEY);
 MOVECOL(TUBEX+WIDTH,TUBEY+SIZE,NONE);
END; (* DRAWTTUBE *)

(*=================================================================*)
PROCEDURE FILLTTUBE(TUBEX,TUBEY,WIDTH,LEVEL:INTEGER;COL:SCREENCOLOR);
(*=================================================================*)
VAR X1,X2,Y:INTEGER;
BEGIN
 PAINT(TUBEX+(WIDTH DIV 2),TUBEY-2,TUBEY,COL);
 X1:=TUBEX+1; X2:=TUBEX+WIDTH-2;
 Y:=TUBEY;
```

```
 REPEAT
  DRAWLINE(X1,Y,X2,Y,COL);
  WAIT(5);
  Y:=Y+1;
 UNTIL Y>LEVEL;
 PENCOLOR(NONE);
END; (* FILLTUBE *)

(*=============================================================*)
PROCEDURE INITMETAL;
(*=============================================================*)
(* set up shapes of metal *)

  (*----------------------------------------------------------*)
  PROCEDURE INITBITS(ROW:INTEGER;VAR BITS:BITSHAPE;S:STRING);
  (*----------------------------------------------------------*)
  VAR COL:INTEGER;
  BEGIN
   FOR COL:=0 TO 15 DO  BITS[ROW,COL]:=S[COL+1]='X';
  END; (* INITBIT *)

BEGIN (* INITMETAL *)
 INITBITS(5,METAL,' XXX    XXXXXX  ');
 INITBITS(4,METAL,' XXXX XXXXXXXXXX');
 INITBITS(3,METAL,'XXXXXXXXXXXXXXXX');
 INITBITS(2,METAL,'XXXXXXXXXXXXXXXX');
 INITBITS(1,METAL,'XXXXXXXXXXXXXXXX');
 INITBITS(0,METAL,' XXXXXXXX  XXXX ');
END; (* INITMETAL *)

(*=============================================================*)
PROCEDURE DROPMETAL(X,Y,BOTTOM:INTEGER);
(*=============================================================*)
BEGIN
 DRAWBLOCK(METAL,2,0,0,16,6,X,Y,MODE); (*display *)
 REPEAT
   DRAWBLOCK(METAL,2,0,0,16,6,X,Y,MODE); (*erase  *)
   Y:=Y-10;
   DRAWBLOCK(METAL,2,0,0,16,6,X,Y,MODE); (*display *)
   WAIT(15);
 UNTIL Y<=(BOTTOM+4);
END;

(*=============================================================*)
PROCEDURE INITBUBBLE;
(*=============================================================*)
  (*----------------------------------------------------------*)
  PROCEDURE INIT(ROW:INTEGER;VAR BITS:BUBSHAPE;S:STRING);
  (*----------------------------------------------------------*)
  VAR COL:INTEGER;
  BEGIN
   FOR COL:=0 TO 7 DO  BITS[ROW,COL]:=S[COL+1]='X';
  END; (* INITBIT *)
```

```
BEGIN (* INITBUBBLE *)
 INIT(5,BUBBLE,'  XX  ');
 INIT(4,BUBBLE,' X X ');
 INIT(3,BUBBLE,'X   X');
 INIT(2,BUBBLE,'X   X');
 INIT(1,BUBBLE,' X X ');
 INIT(0,BUBBLE,'  XX  ');
END; (* INITBUBBLE *)

(*==============================================================*)
PROCEDURE DRAWBUBBLES(X,Y:INTEGER);
(*==============================================================*)
   (*-----------------------------------------------------------*)
   PROCEDURE EXTRABUBBLES(X,NEWY:INTEGER);
   (*-----------------------------------------------------------*)
   VAR MORE:BOOLEAN;
   BEGIN
    REPEAT
     NEWY:=NEWY-GAP;
     MORE:=NEWY>TTUBEY;
     IF MORE  THEN DRAWBLOCK(BUBBLE,2,0,0,8,6,X,NEWY,6)
    UNTIL NOT MORE;
   END; (* EXTRABUBBLES *)

BEGIN (* DRAWBUBBLES *)
 DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE);
 EXTRABUBBLES(X,Y);
END; (* DRAWBUBBLES *)

(*==============================================================*)
PROCEDURE MOVEBUBBLES(VAR X,TOPY:INTEGER);
(*==============================================================*)
VAR Y:INTEGER;
BEGIN (* MOVEBUBBLES *)
 Y:=TOPY;
 TOPY:=TOPY+DY;
 IF TOPY>TTLEVEL THEN TOPY:=TOPY-GAP;
 REPEAT
  DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE); (* erase *)
  IF (Y+DY)<TTLEVEL THEN
     DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE); (*display*)
  Y:=Y-GAP;
 UNTIL Y<TTUBEY;
END; (* MOVEBUBBLES *)

(*==============================================================*)
PROCEDURE DRAWMETAL(X,Y:INTEGER);
(*==============================================================*)
BEGIN
  DRAWBLOCK(METAL,2,0,0,16,6,X,Y,MODE);
END;
```

```
  (*==============================================================*)
  PROCEDURE HEADING;
  (*==============================================================*)
  CONST X=130;
  VAR Y:INTEGER;
  BEGIN
    Y:=120;
    WSTAT(X,Y,'CHEMICAL REACTIONS'); Y:=Y-70;
    WSTAT(X,Y,'Chemistry Dept.'); Y:=Y-15;
    WSTAT(X,Y,'Wollongong Uni.');
  END; (* HEADING *)

BEGIN (* TITLE *)
 INITTURTLE;
 BUBBLX:=TTUBEX+(TTWIDTH DIV 2)-4;
 BUBBLY:=TTUBEY+12;
 DY:=4;SKIP:=4; GAP:=SKIP*DY;
 INITBUBBLE;
 INITMETAL;
 BORDER(VIOLET);
 DRAWTTUBE(TTUBEX,TTUBEY,TTWIDTH,TTSIZE,TTLEVEL,WHITE2);
 FILLTTUBE(TTUBEX,TTUBEY,TTWIDTH,TTLEVEL,WHITE2);
 HEADING;
 DROPMETAL(TTUBEX+8,YMAX-28,TTUBEY);
 DRAWBUBBLES(BUBBLX,BUBBLY);
 COUNT:=0;
 SPEED:=10;
 REPEAT
   COUNT:=COUNT+1;
   MOVEBUBBLES(BUBBLX,BUBBLY);
   WAIT(SPEED);
 UNTIL (COUNT>90) OR (KEYIN);
 DRAWBUBBLES(BUBBLX,BUBBLY); (*erase*)
END; (* TITLE *)

(**************************************************************)
PROCEDURE GETCOLOUR;
(**************************************************************)
CONST XX=50; YY=80;
VAR  X,Y: INTEGER; CH:CHAR;  MONITOR:STRING;
BEGIN
  X:=XMIN+XX; Y:=YMAX-YY;
  WSTAT(X,Y,'Are you using a ');
  WSTAT(X,Y-20,'colour monitor(y/n)');
  GETHICHAR(200,Y-20,CH,['Y','N']);
  IF CH='Y' THEN MONITOR:='INCOL' ELSE MONITOR:='NOCOL';
  SETCVAL(MONITOR);
END; (* GETCOLOUR *)

(**************************************************************)
PROCEDURE INFORM;
(**************************************************************)
CONST XX=30; YY=40;
VAR  X,Y: INTEGER; CH:CHAR;
```

```
(*==============================================================*)
PROCEDURE GETSPACE(TIME:INTEGER);
(*==============================================================*)
VAR COUNT,J :INTEGER;
BEGIN
 CHARTYPE(6);
 COUNT:=0;
 REPEAT
  COUNT:=COUNT+1;
  IF COUNT=TIME THEN
    FOR J:=1 TO 5 DO
     BEGIN
      WSTAT(70,20,'Press <SPACE BAR>');
      WAIT(8);
     END;
 UNTIL (COUNT>TIME) OR (KEYIN);
 CHARTYPE(10);
 GETACHAR(CH,[SPACE,'Q']);
END; (* GETSPACE *)

(*==============================================================*)
PROCEDURE HOWTOQUIT;
(*==============================================================*)
BEGIN
  X:=50; Y:=110;
  WSTAT(X,Y,'To exit this program'); Y:=Y-20;
  WSTAT(X,Y,'at any time input "Q" ');
  GETSPACE(1200);
  QUIT:=CH='Q';
END; (* HOWTOQUIT *)

BEGIN (* INFORM *)
  FILLBOX(15,XMAX-15,15,YMAX-15,BLACK1);
  X:=XMIN+XX; Y:=YMAX-YY;
  WSTAT(X,Y,'IMPORTANT:-'); Y:=Y-45;
  WSTAT(X,Y,'It is necessary to press the'); Y:=Y-25;
  WSTAT(X,Y,'<SPACE BAR> to progress through'); Y:=Y-25;
  WSTAT(X,Y,'this series of programs.');
  GETSPACE(1600);
  QUIT:=CH='Q';
  IF QUIT THEN EXIT(INFORM);

  FILLBOX(15,XMAX-15,15,YMAX-15,BLACK1);
  X:=XMIN+XX; Y:=YMAX-YY;
  WSTAT(X,Y,'This program consists of'); Y:=Y-20;
  WSTAT(X,Y,'four demonstrations.'); Y:=Y-40;
  WSTAT(X,Y,'Each demonstration provides'); Y:=Y-20;
  WSTAT(X,Y,'a macroscopic and microscopic'); Y:=Y-20;
  WSTAT(X,Y,'view of a chemical reaction.');
  GETSPACE(2000);
  QUIT:=CH='Q';
  IF QUIT THEN EXIT(INFORM);
  FILLBOX(15,XMAX-15,15,YMAX-15,BLACK1);
  HOWTOQUIT;
END; (* INFORM *)
```

```
(***********************************************************)
PROCEDURE FIN;
(***********************************************************)
VAR X,Y : INTEGER;
BEGIN
   TEXTMODE;
   PAGE(OUTPUT);
   X:=5; Y:=6;
   WRITE(AT(X+2,Y),'REMOVE DISK FROM DISK DRIVE');Y:=Y+5;
   WRITE(AT(X,Y),'IT WILL BE NECESSARY TO REBOOT'); Y:=Y+2;
   WRITE(AT(X,Y),'COMPUTER TO RUN ANOTHER PROGRAM   ');
   REPEAT
      X:=Y;
   UNTIL X>Y;
END; (* FIN *)

(***********************************************************)
PROCEDURE CHAINTOMENU;
(***********************************************************)
BEGIN
  SETCHAIN(':MENU');
  TEXTMODE;
  WRITE(AT(12,8),'L O A D I N G');
  WRITE(AT(9,12),'M A I N   M E N U          ');
END;

BEGIN (* MAIN *)
   TITLE;
   ENCLOSE(VIOLET);
   GETCOLOUR;
   INFORM;
   IF QUIT THEN FIN ELSE CHAINTOMENU;
END. (* INTRO*)
```

```
(*$S++*)
PROGRAM MENU;

USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;
CONST  STAR='*';
VAR   PROGNUM: CHAR;
      QUIT: BOOLEAN;

 (************************************************************)
 PROCEDURE CHAINTO(VAR ANUM:CHAR);
 (************************************************************)
   (*=========================================================*)
   PROCEDURE INFORM;
   (*=========================================================*)
   BEGIN
    PAGE(OUTPUT);
    WRITE(AT(5,10),'L O A D I N G   P R O G R A M ');
    WRITE(AT(10,15),'PLEASE BE PATIENT ...... ');
   END;

  BEGIN  (* CHAINTO *)
   INFORM;
   CASE ANUM OF
     '1': SETCHAIN(':METAL');
     '2': SETCHAIN(':ACTIVEMETAL');
     '3': SETCHAIN(':CARBONATE');
     '4': SETCHAIN(':LITMUS');
    END; (*CASE*)
  END; (* CHAINTO *)

 (************************************************************)
 PROCEDURE SHOWMENU(VAR ANUM:CHAR);
 (************************************************************)
 CONST DOTS='  .....(';  OFFSET=28;
 VAR  X,Y:INTEGER;
 BEGIN
  PAGE(OUTPUT);
  X:=0;Y:=2;
  WRITE(AT(X,Y),AROW(40,STAR));   Y:=Y+2;
  WRITE(AT(14,Y),'MAIN MENU');     Y:=Y+2;
  WRITE(AT(X,Y),AROW(40,STAR));   Y:=Y+3;
  WRITE(AT(X,Y),'Acid + active metal  ');
  WRITE(AT(OFFSET,Y),CONCAT(DOTS,'1)')); Y:=Y+2;
  WRITE(AT(X,Y),'Water + very active metal');
  WRITE(AT(OFFSET,Y),CONCAT(DOTS,'2)')); Y:=Y+2;
  WRITE(AT(X,Y),'Acid + carbonate       ');
  WRITE(AT(OFFSET,Y),CONCAT(DOTS,'3)')); Y:=Y+2;
  WRITE(AT(X,Y),'Litmus                 ');
  WRITE(AT(OFFSET,Y),CONCAT(DOTS,'4)')); Y:=Y+2;
  WRITE(AT(X,Y),'QUIT                   ');
  WRITE(AT(OFFSET,Y),CONCAT(DOTS,'Q)')); Y:=Y+3;
  WRITE(AT(X+7,Y),'SELECT OPTION       .......(  )');
  GETTEXTCHAR(37,Y,ANUM,['1'..'4','Q']);
  QUIT:=(ANUM='Q');
  PAGE(OUTPUT);
 END;  (* SHOWMENU *)
```

```
(*************************************************************)
PROCEDURE FIN;
(*************************************************************)
VAR  X,Y: INTEGER;
BEGIN
 PAGE(OUTPUT);
 X:=5; Y:=6;
 WRITE(AT(X+2,Y),'REMOVE DISK FROM DISK DRIVE'); Y:=Y+5;
 WRITE(AT(X,Y),'IT WILL BE NECESSARY TO REBOOT'); Y:=Y+2;
 WRITE(AT(X,Y),'COMPUTER TO RUN ANOTHER PROGRAM    ');
 REPEAT
     X:=Y;    (*INFINITE LOOP*)
 UNTIL X>Y;
END;

BEGIN (* MAIN *)
  PAGE(OUTPUT);
  SWAPGPON;          (* set swapping to level 2 *)
  SHOWMENU(PROGNUM);
  IF NOT QUIT THEN CHAINTO(PROGNUM) ELSE FIN;
END. (*MENU*)
```

```
(*$S++*)
(* Microscopic & macroscopic demonstration of reaction between a  reactive metal and
an acid *)

PROGRAM METAL;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;

CONST   MODE=6;
TYPE
  BIGSHAPE=PACKED ARRAY[1..32,1..32] OF BOOLEAN;
  MOLECULE=RECORD
             X,Y,DX,DY : INTEGER;
             SHAPE: BIGSHAPE;
           END;
  PH=(NEUTRAL,ACIDIC,BASIC);

VAR
  ACIDMOL,                                  (*shape & position of 2 acid molecules*)
  WATERMOL: ARRAY[1..2] OF MOLECULE;          (*shape & position ofcorresponding
                                                              watermolecules*)
  ATOM,                                                (*shape of metal atoms*)
  BLANK: BIGSHAPE;
  QUIT: BOOLEAN;
  OPTION: CHAR;

(**********************************************************)
PROCEDURE GETKEY(VAR ACH:CHAR; LEGALSET:CHARSET);
(**********************************************************)
BEGIN
  GETACHAR(ACH,LEGALSET);
  QUIT:=(ACH='Q');
END; (* GETKEY *)

(**********************************************************)
PROCEDURE FALSEARRAY(VAR NEWARRAY:BIGSHAPE);
(**********************************************************)
CONST MAX=32;
VAR  ROW,COL: INTEGER;
BEGIN
  FOR ROW:=1 TO MAX DO
    FOR COL:=1 TO MAX DO  NEWARRAY[ROW,COL]:=FALSE;
END; (* FALSEARRAY *)

(**********************************************************)
PROCEDURE  DEFINESHAPE(VAR NEWARRAY:BIGSHAPE; ACIDITY:PH; ANUM:INTEGER);
(**********************************************************)
(* Defines shape of acid & water molecules   2 orientations (1 & 2 determined by 'anum')
of each   molecule is available.   The array, 'NEWARRAY' must first be initialized to
ALL FALSE!  *)
    VAR CH:CHAR;

    (*===========================================================*)
     PROCEDURE INIT(WIDTH,HEIGHT:INTEGER; SYMBOL:CHAR);
    (*===========================================================*)
    VAR MAXCOL:INTEGER;
```

```
(*---------------------------------------------------------------*)
PROCEDURE MERGE(ROW :INTEGER; S :STRING);
(*---------------------------------------------------------------*)
VAR  COL: INTEGER;
BEGIN
 FOR COL :=1 TO MAXCOL DO
    NEWARRAY[ROW+HEIGHT,COL+WIDTH] :=(S[COL]='X');
END; (* MERGE *)

BEGIN (* INIT *)
 CASE SYMBOL OF
 'H': BEGIN
      MAXCOL :=5;
      MERGE(5, 'X   X');
      MERGE(4, 'X   X');
      MERGE(3, 'XXXXX');
      MERGE(2, 'X   X');
      MERGE(1, 'X   X');
    END; (* INITH *)
 '0':BEGIN
      MAXCOL :=9;
     MERGE(8, '  XXXXX  ');
     MERGE(7, ' X     X ');
     MERGE(6, 'X       X');
     MERGE(5, 'X       X');
     MERGE(4, 'X       X');
     MERGE(3, 'X       X');
     MERGE(2, ' X     X ');
     MERGE(1, '  XXXXX  ');
      END;
 '+': BEGIN
      MAXCOL :=5;
     MERGE(5, '  X  ');
     MERGE(4, '  X  ');
     MERGE(3, 'XXXXX');
     MERGE(2, '  X  ');
     MERGE(1, '  X  ');
    END;
 '1': BEGIN
      MAXCOL :=4;
     MERGE(1, 'XXXX');
    END;
 '2': BEGIN
      MAXCOL :=3;
     MERGE(3, '  X');
     MERGE(2, ' X ');
     MERGE(1, 'X  ');
    END;
 '3': BEGIN
      MAXCOL :=3;
     MERGE(3, 'X  ');
     MERGE(2, ' X ');
     MERGE(1, '  X');
    END;
  END;(*CASE*)
 END; (* INIT *)
```

```
BEGIN (* DEFINESHAPE *)
  CASE ACIDITY OF
   ACIDIC : CASE ANUM OF
        1 : BEGIN
            INIT(12,12,'O');
            INIT(27,0,'H'); (*proton removed in reaction*)
            INIT(27,27,'H');  INIT(0,13,'H');
            INIT(27,14,'+');
            INIT(6,15,'1'); INIT(22,23,'2');
            INIT(22,6,'3'); (*bond broken in reaction *)
           END;
        2: BEGIN
            INIT(9,12,'O');
            INIT(0,0,'H');  (*proton donated in reaction*)
            INIT(0,27,'H');  INIT(27,13,'H');
            INIT(0,14,'+');
            INIT(22,15,'1');
            INIT(6,6,'2');  (*bond broken*)
            INIT(6,22,'3');
           END;
          END; (*ACIDIC*)
    NEUTRAL : CASE ANUM OF
         1 : BEGIN
             (* all y ordinates will be 10 less than in  corresponding acid shape *)
             INIT(12,2,'O');
             INIT(0,3,'H');  INIT(27,17,'H');
             INIT(6,5,'1');  INIT(22,13,'2');
            END;
         2: BEGIN
             INIT(9,2,'O');
             INIT(0,17,'H');  INIT(27,3,'H');
             INIT(22,5,'1');  INIT(6,12,'3');
            END;
         3: BEGIN
             INIT(14,8,'O');
             INIT(0,0,'H');
             INIT(0,16,'H');
             INIT(8,4,'2');
             INIT(8,15,'3');
            END;
         4: BEGIN
             INIT(0,8,'O');
             INIT(16,0,'H');
             INIT(16,16,'H');
             INIT(10,15,'2');
             INIT(10,4,'3');
            END;
         5: BEGIN
             INIT(8,0,'O');
             INIT(0,16,'H');
             INIT(18,16,'H');
             INIT(14,10,'2');
             INIT(7,10,'3');
            END;
```

```
        6: BEGIN
             INIT(8,16,'0');
             INIT(0,3,'H');
             INIT(16,3,'H');
             INIT(5,9,'2');
             INIT(14,9,'3');
           END;
        END; (*CASE*)
   END; (*CASEOF ACIDITY*)
END; (* DEFINESHAPE *)

(************************************************************)
PROCEDURE INITACID;
(************************************************************)
VAR  J: INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
  BEGIN
   WITH ACIDMOL[J] DO BEGIN  SHAPE:=BLANK; DEFINESHAPE(SHAPE,ACIDIC,J);  END;
  WITH WATERMOL[J] DO BEGIN SHAPE:=BLANK; DEFINESHAPE(SHAPE,NEUTRAL,J); END;
   END; (* FOR *)
END;  (* INITACID *)

(************************************************************)
PROCEDURE INITMG;
(************************************************************)
VAR  STR: ARRAY[1..32] OF STRING;  J:INTEGER;

     (*=======================================================*)
     PROCEDURE INIT(ROW:INTEGER;  VAR BITS:BIGSHAPE; S:STRING);
     (*=======================================================*)
     VAR  COL: INTEGER;
     BEGIN
       FOR COL:=1 TO 32 DO  BITS[ROW,COL]:=S[COL]='X';
     END;  (* INIT*)

BEGIN  (* INITMG *)
     STR[1]:='            XXXXXXXX            ';
     STR[2]:='        XXXXXXXXXXXXXXX         ';
     STR[3]:='      XXXXXXXXXXXXXXXXXXX       ';
     STR[4]:='     XXXXXXXXXXXXXXXXXXXXX      ';
     STR[5]:='    XXXXXXXXXXXXXXXXXXXXXXX     ';
     STR[6]:='   XXXXXXXXXXXXXXXXXXXXXXXXX    ';
     STR[7]:='   XXXXXXXXXXXXXXXXXXXXXXXXXX   ';
     STR[8]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXX   ';
     STR[9]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
    STR[10]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
    STR[11]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
    STR[12]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
    STR[13]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
    STR[14]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    STR[15]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    STR[16]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
   FOR J:=1 TO 16 DO
     BEGIN
       INIT(J,ATOM,STR[J]);  INIT(33-J,ATOM,STR[J]);
     END;
END;  (* INITMG *)
```

```
(**********************************************************)
FUNCTION FIN(STR:STRING):BOOLEAN;
(**********************************************************)
 VAR CH:CHAR;
 BEGIN
   INITTURTLE;
   WSTAT(30,100,CONCAT('Repeat ',STR,'SCOPIC'));
   WSTAT(30,70,'demonstration? (Y/N)');
   CHARTYPE(10);
   GETHICHAR(180,70,CH,['N','Y','Q']);
   CHARTYPE(MODE);
   FIN:=CH<>'Y';
   QUIT:=((CH='Q') OR (CH='q'));
 END; (* FIN*)

(**********************************************************)
PROCEDURE MICRO;
(**********************************************************)
TYPE  MEDSHAPE=PACKED ARRAY[1..16,1..16] OF BOOLEAN;  IONTYPE=(MG,NI,MET);
VAR METEL:IONTYPE;  BLANKION,           (*for initialization of cations *)
      CATION:MEDSHAPE;                  (* shape of metal cations *)
      HATOM:PACKED ARRAY[1..5,1..5]OF BOOLEAN;  (*shape of H atom *)
      HYDROGEN:PACKED ARRAY[1..5,1..16]OF BOOLEAN;  (* shape of H2 *)

   (*=======================================================*)
   PROCEDURE DRAWARROW(X,Y,SIZE,TIP:INTEGER);
   (*=======================================================*)
   BEGIN
     MOVECOL(X,Y,WHITE1);
     MOVECOL(X+SIZE,Y,NONE);
     MOVECOL(X+SIZE-TIP,Y+TIP,WHITE1);
     MOVETO(X+SIZE,Y);
     MOVECOL(X+SIZE-TIP,Y-TIP,NONE);
   END;  (* DRAWARROW *)

   (*=======================================================*)
   PROCEDURE INITION(METEL:IONTYPE; VAR ANYION: MEDSHAPE);
   (*=======================================================*)
      (*----------------------------------------------------*)
      PROCEDURE MERGEBITS(ROW:INTEGER; VAR BITS:MEDSHAPE; S:STRING);
      (*----------------------------------------------------*)
      VAR COL:INTEGER;
      BEGIN
         FOR COL:=1 TO 16 DO  BITS[ROW,COL]:=S[COL]='X';
      END; (* MERGEBITS *)

   BEGIN (*INITION *)
     ANYION:=BLANKION;
     CASE METEL OF
       MG: BEGIN
             MERGEBITS(10,ANYION,'X    XX    XXXXXXX');
             MERGEBITS(9,ANYION,'X          X      X');
             MERGEBITS(8,ANYION,'X  X  X  X  X  X');
             MERGEBITS(7,ANYION,'X  X  X  X      X');
             MERGEBITS(6,ANYION,'X  XXXX  XXXX  X');
             MERGEBITS(5,ANYION,' XXXXXXXXX     X ');
         END;
```

```
NI: BEGIN
      MERGEBITS(10,ANYION,'X   XXX  XXXXXXX');
      MERGEBITS(9,ANYION,'X    XX  XX  XXX');
      MERGEBITS(8,ANYION,'X     X  XXXXXXX');
      MERGEBITS(7,ANYION,'X  X     XX  XXX');
      MERGEBITS(6,ANYION,'X  XXX   XX  XXX');
  END;
MET: BEGIN
      MERGEBITS(9,ANYION,'XXXX   XX   XXXX');
      MERGEBITS(8,ANYION,'XXXX        XXXX');
      MERGEBITS(7,ANYION,'XXXX  X  X  XXXX');
      MERGEBITS(6,ANYION,'XXXX  X  X  XXXX');
      MERGEBITS(5,ANYION,' XXX  XXXX  XXX ');
  END;
 END;(*CASE*)
END; (* INITION *)

(*==========================================================*)
PROCEDURE INITSHAPES;
(*==========================================================*)
  (*--------------------------------------------------------*)
  PROCEDURE INITBLANK(VAR ANYION:MEDSHAPE);
  (*--------------------------------------------------------*)
    (*......................................................*)
    PROCEDURE INITBITS(ROW:INTEGER; VAR BITS:MEDSHAPE; S:STRING);
    (*......................................................*)
    VAR  COL:INTEGER;
    BEGIN
      FOR COL:=1 TO 16 DO  BITS[ROW,COL]:=S[COL]='X';
    END; (* INITBITS *)

  BEGIN (* INITBLANK *)
   INITBITS(16,ANYION,'      XXXX      ');
   INITBITS(15,ANYION,'   X  XXX  X    ');
   INITBITS(14,ANYION,'  XX  XXX  XXX  ');
   INITBITS(13,ANYION,' X     X     XX ');
   INITBITS(12,ANYION,' XXX  XXX  XXXX ');
   INITBITS(11,ANYION,'XXXX  XXX  XXXXX');
   INITBITS(10,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(9,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(8,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(7,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(6,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(5,ANYION,' XXXXXXXXXXXXXX ');
    INITBITS(4,ANYION,'  XXXXXXXXXXXX  ');
    INITBITS(3,ANYION,'  XXXXXXXXXXXX  ');
    INITBITS(2,ANYION,'   XXXXXXXXXX   ');
    INITBITS(1,ANYION,'     XXXXXX     ');
  END; (* INITBLANK *)

  (*--------------------------------------------------------*)
  PROCEDURE INITHATOM;
  (*--------------------------------------------------------*)
    (*......................................................*)
    PROCEDURE INIT(ROW:INTEGER; S:STRING);
    (*......................................................*)
    VAR  COL: INTEGER;
    BEGIN   FOR COL:=1 TO 5 DO  HATOM[ROW,COL]:=S[COL]='X';  END; (*INIT*)
```

```
BEGIN (* INITHATOM *)
   INIT(5,'X   X');
   INIT(4,'X   X');
   INIT(3,'XXXXX');
   INIT(2,'X   X');
   INIT(1,'X   X');
END; (* INITHATOM *)

(*--------------------------------------------------------------------*)
PROCEDURE INITHYDROGEN;
(*--------------------------------------------------------------------*)

   (*...................................................................*)
   PROCEDURE INIT(ROW:INTEGER; S:STRING);
   (*...................................................................*)
   VAR COL:INTEGER;
   BEGIN
    FOR COL:=1 TO 16 DO  HYDROGEN[ROW,COL]:=S[COL]='X';
   END; (* INIT *)

 BEGIN (* INITHYDROGEN *)
    INIT(5,'X   X    X   X');
    INIT(4,'X   X    X   X');
    INIT(3,'XXXXX XXXX XXXXX');
    INIT(2,'X   X    X   X');
    INIT(1,'X   X    X   X');
 END; (* INITHYDROGEN *)


BEGIN (*INITSHAPES*)
   INITBLANK(BLANKION);
   INITION(MG,CATION);
   INITHATOM;
   INITHYDROGEN;
END; (* INITSHAPES *)


(*$I :METAL2*)
(*$I :METAL3*)
(*$I :METAL4*)
(*$I :METAL5*)
(*$I :METAL6*)
(*$I :METAL7*)
```

```
(* METAL2 *)
  (*=================================================================*)
  PROCEDURE EXPLAINSHAPES;
  (*=================================================================*)

    (*----------------------------------------------------------------*)
    PROCEDURE SHOWSHAPES;
    (*----------------------------------------------------------------*)
     VAR CH:CHAR;
     BEGIN
       INITTURTLE;
       WSTAT(0,172,'    This demonstration will display');
       WSTAT(0,152,'following structures:-');
       DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,140,110,MODE);
       WSTAT(0,110,'WATER MOLECULE:');
       DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,140,45,MODE);
       WSTAT(0,45,'HYDRONIUM ION:');
       GETKEY(CH,[SPACE,'Q']);
       IF QUIT THEN EXIT(MICRO);
     END; (* SHOWSHAPES *)

     (*---------------------------------------------------------------*)
     PROCEDURE ATOMSTRUCTURE;
     (*---------------------------------------------------------------*)
     VAR METALX,METALY : INTEGER; CH:CHAR;

        (*............................................................*)
        PROCEDURE SHOWMETAL(X,Y:INTEGER);
        (*............................................................*)
        BEGIN
          DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
          WSTAT(X,Y-12,'Magnesium');
          WSTAT(X,Y-21,' Atom ');
          WSTAT(50,Y-70,'A magnesium atom has');
          WSTAT(50,Y-80,'2 valence electrons.');
        END; (* SHOWMETAL *)

        (*............................................................*)
        PROCEDURE SHOWSTRUCTURE(X,Y:INTEGER);
        (*............................................................*)

          PROCEDURE ARROW(X,Y,SIZE:INTEGER);
          (*this arrow pts in opposite direction to DRAWARROW *)
          CONST TIP=3;
          BEGIN
            MOVECOL(X-SIZE+TIP,Y+TIP,WHITE2);
            MOVETO(X-SIZE,Y);
            MOVECOL(X-SIZE+TIP,Y-TIP,NONE);
            MOVECOL(X-SIZE+1,Y,WHITE2);
            MOVETO(X-SIZE,Y);
            MOVECOL(X,Y,NONE);
          END; (*ARROW*)
```

```
    BEGIN (* SHOWSTRUCTURE *)
      DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE); (*display Metal ion*)
      WSTAT(X+12,Y+26,'e');
      WSTAT(X+12,Y,'e');
      ARROW(X+40,Y+30,8);
      ARROW(X+40,Y+15,6);
      ARROW(X+40,Y+4,6);
      WSTAT(X+45,Y+26,'valence electron');
      WSTAT(X+45,Y,'valence electron');
      WSTAT(X+45,Y+11,'positive ion');
    END; (* SHOWSTRUCTURE *)

  BEGIN (* ATOMSTRUCTURE *)
    INITTURTLE;
    METALX:=110; METALY:=115;
    SHOWMETAL(METALX,METALY);
    GETKEY(CH,[SPACE,'Q']);
    IF QUIT THEN EXIT(MICRO);
    SHOWSTRUCTURE(METALX,METALY);
    GETKEY(CH,[SPACE,'Q']);
  END; (* ATOMSTRUCTURE *)

  (*-------------------------------------------------------------------*)
  PROCEDURE METALSTRUCTURE;
  (*-------------------------------------------------------------------*)
  CONST STARTX=80; STARTY=60; NUMSTR=4;
  TYPE MANYSTR=ARRAY[1..NUMSTR] OF STRING;
  VAR STR:MANYSTR;

     (*...............................................................*)
     PROCEDURE DRAWATOMS(X1,Y1:INTEGER);
     (*...............................................................*)
     CONST SIZE=32;
     TYPE OUTLINE=(OUTER,INNER);
     VAR  SHAPE:OUTLINE;
        X,Y:INTEGER;CH:CHAR;

        PROCEDURE DRAWAROW(XX,YY,ANYNUM:INTEGER);
        VAR  I: INTEGER;
        BEGIN
         FOR I:=1 TO ANYNUM DO
         BEGIN
          IF SHAPE=OUTER THEN DRAWBLOCK(ATOM,4,0,0,32,32,XX,YY,MODE)
          ELSE  DRAWBLOCK(CATION,2,0,0,16,16,XX+8,YY+8,MODE);
          YY:=YY+SIZE;
         END;
        END; (* DRAWAROW *)

     BEGIN (* DRAWATOMS *)
      FOR SHAPE:=OUTER TO INNER DO
       BEGIN
        X:=X1; Y:=Y1;
        DRAWAROW(X,Y,2);
        X:=X+SIZE-5;
        DRAWAROW(X,Y-(SIZE DIV 2),3);
        X:=X+SIZE-5;
```

```
    DRAWAROW(X,Y,2);
    IF SHAPE=OUTER THEN GETKEY(CH,[SPACE,'Q']);
    IF QUIT THEN EXIT(METALSTRUCTURE);
  END;
END; (* DRAWATOMS *)

(*.....................................................................................................*)
PROCEDURE SHOWTEXT(VAR S:MANYSTR);
(*.....................................................................................................*)
CONST X=20;
VAR Y,J:INTEGER;
BEGIN
 Y:=182;
 FOR J:=1 TO NUMSTR DO
   BEGIN  WSTAT(X,Y,S[J]); Y:=Y-12;  END;
END; (* SHOWTEXT *)

(*.....................................................................................................*)
PROCEDURE CRYSTAL(X1,Y1:INTEGER);
(*.....................................................................................................*)
BEGIN
  STR[1]:='Magnesium atoms are arranged in a';
  STR[2]:='crystal structure in which each';
  STR[3]:='magnesium atom is surrounded by';
  STR[4]:='many other magnesium atoms.';
  SHOWTEXT(STR); (*DISPLAY TEXT*)
  DRAWATOMS(X1,Y1);
  FILLAREA(XMIN,XMAX,140,YMAX,BLACK1); (*ERASE TEXT*)
END; (* CRYSTAL *)

(*.....................................................................................................*)
PROCEDURE BONDING;
(*.....................................................................................................*)
VAR CH:CHAR;
BEGIN
  STR[1]:=' Metallic bonding is often';
  STR[2]:=' described as positive';
  STR[3]:=' metal ions embedded';
  STR[4]:=' in an "electronic glue".';
  SHOWTEXT(STR); (* DISPLAY TEXT *)
  GETKEY(CH,[SPACE,'Q']);
  FILLAREA(XMIN,XMAX,140,YMAX,BLACK1);(*erase text*)
END; (* BONDING *)

(*.....................................................................................................*)
PROCEDURE LOSELECTRONS(X1,Y1:INTEGER);
(*.....................................................................................................*)
VAR ELECY,K : INTEGER;  ELECX:ARRAY[1..2] OF INTEGER; CH:CHAR;

   PROCEDURE FLASHELECTRON(VAR X,Y:INTEGER;  DX:INTEGER);
   BEGIN
     WSTAT(X,Y,'e'); (* erase electron*)
     X:=X-DX;
     DELAY(70);
     WSTAT(X,Y,'e'); (* display electron*)
   END; (* FLASHELECTRON *)
```

```
PROCEDURE REMOVEATOM(X,Y:INTEGER);
CONST DX=8; DY=4;
VAR J: INTEGER;
BEGIN
 DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,4); (*erase*)
 FOR J:=1 TO 7 DO
  BEGIN                  (*display*)
    DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE);
    DELAY(20);
    DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE);
    X:=X-DX; Y:=Y-DY;          (* erase *)
  END; (* FOR*)
 END; (* REMOVEATOM *)

BEGIN (* LOSELECTRONS *)
 STR[1]:='If two electrons are removed from';
 STR[2]:='this crystal structure then a';
 STR[3]:='positive ion is lost from the';
 STR[4]:='crystal structure.';
 SHOWTEXT(STR); (*display text*)
 ELECX[1]:=X1+2;
 ELECX[2]:=X1+26; ELECY:=Y1+12;
 WSTAT(ELECX[1],ELECY,'e'); (* display electrons*)
 WSTAT(ELECX[2],ELECY,'e');
 STR[1]:='Press <SPACE BAR> to remove electrons';
 WSTAT(1,1,STR[1]); (* DISPLAY *)
 CH:='X';
 REPEAT
   FLASHELECTRON(ELECX[1],ELECY,0);
   FLASHELECTRON(ELECX[2],ELECY,0);
   IF KEYIN THEN READ(CH);
 UNTIL (CH=SPACE);
 WSTAT(1,1,STR[1]); (* ERASE  *)
 FOR K:=1 TO 5 DO
  BEGIN
    FLASHELECTRON(ELECX[1],ELECY,4);
    FLASHELECTRON(ELECX[2],ELECY,4);
  END;
 WSTAT(ELECX[1],ELECY,'e'); (* erase electron*)
 WSTAT(ELECX[2],ELECY,'e'); (* erase electron*)
 REMOVEATOM(X1,Y1);
 GETKEY(CH,[SPACE,'Q']);
END; (* LOSELECTRONS *)

BEGIN (* METALSTRUCTURE *)
 INITTURTLE;
 CRYSTAL(STARTX,STARTY);
 BONDING;
 LOSELECTRONS(STARTX,STARTY);
END; (* METALSTRUCTURE *)

BEGIN (* EXPLAINSHAPES *)
 SHOWSHAPES;
 ATOMSTRUCTURE;
 IF NOT QUIT THEN METALSTRUCTURE;
END; (* EXPLAINSHAPES *)
```

```
(* METAL3 *)
   (*=================================================================*)
   PROCEDURE SOLVATECATIONS;
   (*=================================================================*)
   CONST WATERNUM=4;
   VAR WATER: ARRAY[1..WATERNUM] OF BIGSHAPE; (*shape of waters of hydration*)
     AQUA:INTEGER;         (* index to shape of water    *)

     (*-----------------------------------------------------------------*)
     PROCEDURE DEFINEWATER;
     (*-----------------------------------------------------------------*)
     (*Watermolecule shapes numbered 1 - 4 correspond to shapes 3-6 in Defineshape*)
     BEGIN (* DEFINEWATER *)
      WHILE ((AQUA<=WATERNUM) AND (NOT KEYIN)) DO
       BEGIN
         WATER[AQUA]:=BLANK;
         DEFINESHAPE(WATER[AQUA],NEUTRAL,AQUA+2);
         AQUA:=AQUA+1;
       END;
     END; (* DEFINEWATER *)

     (*-----------------------------------------------------------------*)
     PROCEDURE HYDRATE(ANUM,CENTRX,CENTRY:INTEGER);
     (*-----------------------------------------------------------------*)
     VAR SIZE,DISTANCE,RADIUS,BONDLEN, X,Y,X1,Y1,X2,Y2,J :INTEGER;
     BEGIN
       SIZE:=24; DISTANCE:=20;
       BONDLEN:=4; RADIUS:=15;
       FOR J:=1 TO ANUM DO
         BEGIN
          CASE J OF
          1: BEGIN
               X:=CENTRX-DISTANCE-SIZE;
               Y:=CENTRY-(SIZE DIV 2);
               X1:=CENTRX-RADIUS-2; Y1:=CENTRY;
               X2:=X1+BONDLEN; Y2:=Y1;
             END;
          2: BEGIN
               X:=CENTRX+DISTANCE;
               Y:=CENTRY-(SIZE DIV 2);
               X1:=CENTRX+RADIUS; Y1:=CENTRY;
               X2:=X1-BONDLEN; Y2:=Y1;
             END;
          3: BEGIN
               X:=CENTRX-(SIZE DIV 2);
               Y:=CENTRY+DISTANCE;
               X1:=CENTRX-2; Y1:=CENTRY+RADIUS;
               X2:=X1;     Y2:=Y1-BONDLEN;
             END;
          4: BEGIN
               X:=CENTRX-(SIZE DIV 2);
               Y:=CENTRY-DISTANCE-SIZE;
               X1:=CENTRX-2; Y1:=CENTRY-RADIUS;
               X2:=X1;     Y2:=Y1+BONDLEN;
             END;
            END; (*CASE*)
```

```
      DRAWBLOCK(WATER[J],4,0,0,24,24,X,Y,MODE);
      DRAWLINE(X1,Y1,X2,Y2,WHITE2);
    END;
  END; (* HYDRATE *)

(*-----------------------------------------------------------------*)
PROCEDURE INTROSOLVATE;
(*-----------------------------------------------------------------*)
VAR X,Y:INTEGER;CH:CHAR;
BEGIN
  INITTURTLE;
  X:=10; Y:=YMAX-40;
  WSTAT(X,Y,'This demonstration simplifies the'); Y:=Y-20;
  WSTAT(X,Y,'reaction of ions in solution.'); Y:=Y-50;
  WSTAT(X,Y,'In solution ions are SOLVATED.'); Y:=Y-50;
  WSTAT(X,Y,'In aqueous solution ions are HYDRATED.');
  DEFINEWATER; (*while waiting for <space> set up arrays*)
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
END; (* INTROSOLVATE *)

(*-----------------------------------------------------------------*)
PROCEDURE SHOWHYDRATE;
(*-----------------------------------------------------------------*)
CONST ST='Press <SPACE BAR> to show hydrated ion';
VAR MIDX,MIDY : INTEGER; CH:CHAR;
BEGIN
  INITION(MET,CATION);
  INITTURTLE;
  MIDX:=(XMAX DIV 2)-10; MIDY:=(YMAX DIV 2)+30;
  DRAWBLOCK(CATION,2,0,0,16,16,MIDX-8,MIDY-8,MODE);
  WSTAT(XMIN,YMIN+40,'This represents any metal ion.');
  REPEAT
    DEFINEWATER;
    IF KEYIN THEN READ(CH);
  UNTIL AQUA>WATERNUM;
  WSTAT(XMIN,YMIN,ST);
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
  WSTAT(XMIN,YMIN,ST);
  HYDRATE(WATERNUM,MIDX,MIDY);
  WSTAT(XMIN,YMIN+15,'The number of water molecules involved');
  WSTAT(XMIN,YMIN,'in hydration varies for each metal ion.');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
END; (* SHOWHYDRATE *)

(*-----------------------------------------------------------------*)
PROCEDURE NETMETALREACTION;
(*-----------------------------------------------------------------*)
VAR X,Y : INTEGER; CH:CHAR;
BEGIN
  INITTURTLE;
  X:=0; Y:=YMAX-10;
  WSTAT(X,Y,'Net reaction of metal:-'); Y:=Y-40;
  WSTAT(X,Y,'METAL ION'); Y:=Y-10;
```

```
      WSTAT(X,Y,'IN ELECTRONIC');DRAWARROW(X+100,Y+4,20,7);
      WSTAT(X+135,Y,'HYDRATED METAL ION');  Y:=Y-10;
      WSTAT(X,Y,' "GLUE" ');  Y:=Y-50;  X:=X+40;
      DRAWBLOCK(ATOM,4,0,0,32,32,X,Y-16,MODE);
      DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y-8,MODE); X:=X+130;
      DRAWBLOCK(CATION,2,0,0,16,16,X-8,Y-8,MODE);
      HYDRATE(WATERNUM,X,Y);  X:=X+65;
      WSTAT(X,Y+4,'e');
      WSTAT(X,Y-16,'e');
      GETKEY(CH,[SPACE,'Q']);
     END; (* NETMETALREACTION *)

   BEGIN (* SOLVATECATION *)
    AQUA:=1;
    INTROSOLVATE;
    SHOWHYDRATE;
    NETMETALREACTION;
   END; (* SOLVATECATIONS *)


(* METAL4 *)
   (*================================================================*)
   PROCEDURE REACTION;
   (*================================================================*)
   TYPE  ION=RECORD
           X,Y,DX,DY: INTEGER;
          END;
   VAR CYCLE, METALX,METALY :INTEGER;   METALION: ION;

     (*----------------------------------------------------------------*)
     PROCEDURE CHECKKEY;
     (*----------------------------------------------------------------*)
     (*If space bar (or return) then pause program until space
      bar is again pressed to restart program  *)
     VAR CH:CHAR;
     BEGIN
       READ(CH);
       IF CH=SPACE THEN
        BEGIN
          CHARTYPE(10);
          WSTAT(186,182,'continue');
          GETKEY(CH,[SPACE,'Q']);
          WSTAT(186,182,'pause   ');
          CHARTYPE(6);
        END
        ELSE
           QUIT:=((CH='Q') OR (CH='q'));
       IF QUIT THEN EXIT(REACTION);
     END; (* CHECKKEY *)
```

```
(*---------------------------------------------------------------------*)
PROCEDURE DRAWMETAL(SYMBOL:IONTYPE);
(*---------------------------------------------------------------------*)
(*draw 2 rows of metal atoms *)
VAR  J,LASTX,X,Y : INTEGER;  CH:CHAR; SYMB: STRING[2];ASTR:STRING;
BEGIN
 X:=16; Y:=1;
 LASTX:=XMAX-32;
 CASE SYMBOL OF
  MG: BEGIN  SYMB:='Mg';  ASTR:='magnesium';  END;
  NI:  BEGIN  SYMB:='Ni';  ASTR:='nickel';   END;
  END; (*CASE*)
 FOR J:=1 TO 2 DO
  BEGIN
   REPEAT
    DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
    MOVETO(X+8,Y+12);
    WSTRING(SYMB);  X:=X+32;
   UNTIL (X>=LASTX);
   X:=0; Y:=Y+31;
  END;
 WSTAT(0,184,CONCAT('The surface atoms of ',ASTR));
 WSTAT(0,174,'are represented by the structure:-');
 GETKEY(CH,[SPACE,'Q']);
 FILLAREA(0,XMAX,172,YMAX,BLACK1);
END; (* DRAWMETAL *)

(*---------------------------------------------------------------------*)
PROCEDURE NEWVALUE(ANYNUM:INTEGER);
(*---------------------------------------------------------------------*)
(* calculates starting pt of acid based on final  position required to react with
   metal atom *)
CONST INCR=5;
VAR  CENTMGX, TOPMGY,  (*coord. of top centre of target metal atom*)
      ANUM  : INTEGER;   (*ANUM determines which metal atom reacts *)
BEGIN
  CASE ANYNUM OF
  1: BEGIN
      ANUM:=4;
      WITH METALION DO    BEGIN  DX:=6;  DY:=8;    END;
      WITH ACIDMOL[1] DO  BEGIN  DX:=12;  DY:=-12;  END;
      WITH ACIDMOL[2] DO   BEGIN  DX:=-12; DY:=-12;  END;
    END; (*  1   *)
   2: BEGIN
      ANUM:=3;
      WITH METALION DO    BEGIN  DX:=-6;  DY:=8;    END;
      WITH ACIDMOL[1] DO  BEGIN  DX:=12;  DY:=-12;  END;
      WITH ACIDMOL[2] DO  BEGIN  DX:=-12; DY:=-12;  END;
    END;  (* 2 *)
   3: BEGIN
      ANUM:=2;
      WITH METALION DO   BEGIN   DX:=10;  DY:=8;    END;
      WITH ACIDMOL[1] DO BEGIN   DX:=4;    DY:=-12;  END;
      WITH ACIDMOL[2] DO BEGIN   DX:=-12; DY:=-12;  END;
    END; (*  3  *)
  END;(* CASE*)
```

```
  METALX:=32*ANUM; METALY:=32;
  CENTMGX:=METALX+16; TOPMGY:=64;
  WITH ACIDMOL[1] DO
   BEGIN
    X:=CENTMGX-42; (*see DEFINESHAPE - proton removed is 22 from bottom
    Y:=TOPMGY+10;                                          L.H.C.*)
    X:=X-(INCR*DX);
    Y:=Y-(INCR*DY);
   END;
  WITH ACIDMOL[2] DO
   BEGIN
    X:=CENTMGX+10;
    Y:=TOPMGY+10;
    X:=X-(INCR*DX);
    Y:=Y-(INCR*DY);
   END;
 END; (* NEWVALUE *)

(*----------------------------------------------------------------*)
PROCEDURE SHOWACID(TEMPMODE:INTEGER);
(*----------------------------------------------------------------*)
VAR NUM: INTEGER;
BEGIN
 FOR NUM:=1 TO 2 DO
   BEGIN
    WITH ACIDMOL[NUM] DO  DRAWBLOCK(SHAPE,4,0,0,32,32,X,Y,TEMPMODE);
   END;
END; (* SHOWACID *)

(*----------------------------------------------------------------*)
PROCEDURE DISPLAYPROMPT;
(*----------------------------------------------------------------*)
VAR CH: CHAR;
     PROCEDURE PROMPT;
     BEGIN
          WSTAT(0,182,'<SPACE BAR> to react acid with metal');
     END;
 BEGIN
  PROMPT;
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(MICRO);
  PROMPT;
  WSTAT(40,182,'Press <SPACE BAR> to pause');
 END; (* DISPLAYACID *)

(*----------------------------------------------------------------*)
PROCEDURE MOVEMOLECULE(VAR ANYREC:MOLECULE;  HEIGHT:INTEGER);
(*----------------------------------------------------------------*)
BEGIN
 WITH ANYREC DO
  BEGIN
   DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
   X:=X+DX; Y:=Y+DY;
   DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
  END;
END; (* MOVEMOLECULE *)
```

```
(*---------------------------------------------------------------------*)
PROCEDURE MOVEACID(INCR:INTEGER);
(*---------------------------------------------------------------------*)
(* INCR required in NEWVALUE to calc. starting position  of acid*)
CONST HEIGHT=32;
VAR  STEP: INTEGER;
BEGIN
 FOR STEP:=1 TO INCR DO
  BEGIN
    MOVEMOLECULE(ACIDMOL[1],HEIGHT);
    IF KEYIN THEN CHECKKEY;
    MOVEMOLECULE(ACIDMOL[2],HEIGHT);
  END;
END; (* MOVEACID *)

(*---------------------------------------------------------------------*)
PROCEDURE REVERSEACID;
(*---------------------------------------------------------------------*)
VAR NUM:INTEGER;
BEGIN
  FOR NUM:=1 TO 2 DO
   WITH ACIDMOL[NUM] DO
    BEGIN
      DX:=-ACIDMOL[NUM].DX;
      DY:=-ACIDMOL[NUM].DY;
    END;
 END; (* REVERSEACID *)

(*---------------------------------------------------------------------*)
PROCEDURE MOVEWATER(NUM:INTEGER);
(*---------------------------------------------------------------------*)
BEGIN
  MOVEMOLECULE(WATERMOL[NUM],24);
END; (* MOVEWATER *)

(*---------------------------------------------------------------------*)
PROCEDURE REACT;
(*---------------------------------------------------------------------*)
VAR   I,PROTONX,PROTONY : INTEGER;

  (*..................................................................*)
  PROCEDURE POLARISE;
  (*..................................................................*)
  VAR NUM,X1,Y1 : INTEGER;
  BEGIN
   CHARTYPE(10);
   FOR NUM:=1 TO 2 DO
    BEGIN
     Y1:=12;
     CASE NUM OF
      1: X1:=26;
      2: X1:=0;
     END; (*CASE*)
     WITH ACIDMOL[NUM] DO
```

```
      BEGIN
        X1 :=X+X1 ; Y1 :=Y+Y1 ;
        WSTAT(X1 ,Y1 ,' ');   (* erase *)
        WSTAT(X1 ,Y1-6 ,'+'); (* display *)
      END;
    END; (*FOR*)
  CHARTYPE(MODE);
  DELAY(100);
END; (* POLARISE *)

(*...........................................................................................*)
PROCEDURE CALCULATE;
(*...........................................................................................*)
(*initial position of water is same as final position of acid *)
VAR NUM:INTEGER;
BEGIN
  FOR NUM:=1 TO 2 DO
    WITH WATERMOL[NUM] DO
    BEGIN
      X:=ACIDMOL[NUM].X;
      Y:=ACIDMOL[NUM].Y+10;  (*see DEFINESHAPE *)
      DX:=-ACIDMOL[NUM].DX;
      DY:=-ACIDMOL[NUM].DY;
    END;
  PROTONX:=ACIDMOL[1].X+22; (*see DEFINESHAPE*)
  PROTONY:=ACIDMOL[1].Y;
END; (* CALCULATE *)

(*...........................................................................................*)
PROCEDURE IONIZE(VAR METX,METY:INTEGER);
(*...........................................................................................*)
(* exchange metal atom for metal ion & initialize   metal ion record*)
BEGIN
  DRAWBLOCK(ATOM,4,0,0,32,32,METX,METY,4);   (*erase metal atom*)
  WITH METALION DO
    BEGIN
      X:=METX+8; Y:=METY+8;
      DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);  (* display ion*)
    END;
END; (* IONIZE  *)

(*...........................................................................................*)
PROCEDURE SHOWWATER;
(*...........................................................................................*)
VAR NUM : INTEGER;
BEGIN
  FOR NUM:=1 TO 2 DO
    BEGIN
      WITH WATERMOL[NUM] DO DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,MODE);
    END;
END; (* SHOWWATER *)
```

```
(*...........................................................................................................*)
PROCEDURE SHOWPROTONS(X,Y:INTEGER);
(*...........................................................................................................*)
VAR J : INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
  BEGIN
    DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
    WSTAT(X+2,Y+6,'+');
    X:=X+25;
  END;
END; (* SHOWPROTONS *)

(*...........................................................................................................*)
PROCEDURE SHOWELECTRONS(X,Y:INTEGER);
(*...........................................................................................................*)
VAR J: INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
  BEGIN
   WSTAT(X+8,Y-1,'e');
   X:=X+8;
  END;
END;  (* SHOWELECTRONS *)

(*...........................................................................................................*)
PROCEDURE SHOWHYDROGEN(X,Y:INTEGER);
(*...........................................................................................................*)
BEGIN
 DRAWBLOCK(HYDROGEN,2,0,0,16,5,X,Y,MODE);
END;

(*...........................................................................................................*)
PROCEDURE MOVEH2(VAR X,Y:INTEGER);
(*...........................................................................................................*)
BEGIN
 SHOWHYDROGEN(X,Y);  (* erase H2 *)
 Y:=Y+8;
 SHOWHYDROGEN(X,Y);  (* display H2 *)
END; (* MOVEH2 *)

(*...........................................................................................................*)
PROCEDURE MOVEMGION;
(*...........................................................................................................*)
BEGIN
 WITH METALION DO
  BEGIN
   DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);   (*erase*)
   X:=X+DX; Y:=Y+DY;
   DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
  END;
END;  (* MOVEMGION *)
```

```
BEGIN (* REACT *)
  POLARISE; (* show +ve charge closer to metal atoms *)
  CALCULATE; (* coord. of water molecules & direction *)
  IF KEYIN THEN CHECKKEY;
  SHOWACID(0);   (* erase both acid molecules from current position
                    as well as +ve charges in new position *)
  SHOWWATER; (*display 2 water molecules*)
  IONIZE(METALX,METALY); (* replace atom with ion  *)
  SHOWPROTONS(PROTONX,PROTONY);  (*display 2 protons *)
  SHOWELECTRONS(PROTONX,PROTONY); (*display 2 electrons*)
  DELAY(150);
  IF KEYIN THEN CHECKKEY;   (* wait then erase protons & electtrons*)
  SHOWPROTONS(PROTONX,PROTONY);  (*erase 2 protons *)
  SHOWELECTRONS(PROTONX,PROTONY); (*erase 2 electrons*)
  PROTONX:=PROTONX+7; (*H2 is 16 bits wide whereas H+ -H+
    was 30 bits wide therfore move across 7 to centre*)
  SHOWHYDROGEN(PROTONX,PROTONY); (*display H2 molecule*)
  IF KEYIN THEN CHECKKEY;
  FOR I:=1 TO 4 DO
   BEGIN      (*move water molecules & H2 molecule*)
     MOVEWATER(1);
     MOVEWATER(2);
     MOVEH2(PROTONX,PROTONY);
     IF KEYIN THEN CHECKKEY;
   END;
  FOR I:=1 TO 12 DO
   BEGIN   (*move water molecules,H2 molecule & metal ion*)
    IF ODD(I) THEN MOVEWATER(1) ELSE  MOVEWATER(2);
    MOVEH2(PROTONX,PROTONY);
    MOVEMGION;
    IF KEYIN THEN CHECKKEY;
   END;
  SHOWWATER;  (* erase 2 water molecules *)
  SHOWHYDROGEN(PROTONX,PROTONY); (* erase H2 molecule *)
  DRAWBLOCK(CATION,2,0,0,16,16,METALION.X,METALION.Y,MODE);
               (* erase metal ion*)
 END; (* REACT *)


BEGIN  (* REACTION *)
 FOR METEL:=MG TO NI DO
  IF NOT QUIT THEN
   BEGIN
    INITION(METEL,CATION);
    INITTURTLE;
    DRAWMETAL(METEL);
    IF QUIT THEN EXIT(REACTION);
    CYCLE:=0;
    REPEAT
     CYCLE:=CYCLE+1;
     NEWVALUE(CYCLE);
     SHOWACID(MODE);
     IF CYCLE=1 THEN DISPLAYPROMPT;(*start reaction*)
     MOVEACID(5);
     IF ((METEL=NI) AND (CYCLE>1)) THEN
```

```
        BEGIN
          DELAY(400);
          REVERSEACID;
          MOVEACID(10);
          NEWVALUE(CYCLE);
          SHOWACID(MODE);
          MOVEACID(5);
        END;
       IF NOT QUIT THEN REACT;
      UNTIL ((CYCLE=3) OR QUIT);   (* H30+ reacts with 3 metal atoms *)
     END;(* IF *)
   END; (* REACTION *)


(* METAL5 *)
   (*==============================================================*)
   PROCEDURE CONCLUSION;
   (*==============================================================*)

      (*-------------------------------------------------------------*)
      PROCEDURE DRAWPLUS(X,Y:INTEGER);
      (*-------------------------------------------------------------*)
      CONST  SIZE=10;
      BEGIN
       MOVECOL(X,Y,WHITE1);
       MOVECOL(X+SIZE,Y,NONE);
       X:=X+(SIZE DIV 2);
       Y:=Y+(SIZE DIV 2);
       MOVECOL(X,Y,WHITE1);
       MOVECOL(X,Y-SIZE,NONE);
      END; (* DRAWPLUS *)

      (*-------------------------------------------------------------*)
      PROCEDURE DRAWH2(X,Y:INTEGER);
      (*-------------------------------------------------------------*)
      BEGIN
       DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
       DRAWBLOCK(HATOM,2,0,0,5,5,X,Y+15,MODE);
       MOVECOL(X+2,Y+7,WHITE1);
       MOVECOL(X+2,Y+12,NONE);
      END; (* DRAWH2 *)

      (*-------------------------------------------------------------*)
      PROCEDURE NETMETAL;
      (*-------------------------------------------------------------*)
      VAR SYMBOL:STRING[2]; X,Y:INTEGER;
      BEGIN
       Y:=YMAX-10;
       INITTURTLE;
       WSTAT(0,Y,'Net reaction of metal:-');
       Y:=Y-60;
       (*from reaction cation represents Ni 2+ ion*)
       FOR METEL:=MG TO NI DO
```

```
  BEGIN
    INITION(METEL,CATION); (* initialize cation to Mg2+*)
    CASE METEL OF
      MG:SYMBOL:='Mg';
      NI:SYMBOL:='Ni';
      END; (*CASE*)
    X:=1;
    DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
    WSTAT(X+8,Y+12,SYMBOL);  X:=X+70;
    DRAWARROW(X,Y+16,30,7);  X:=X+80;
    DRAWBLOCK(CATION,2,0,0,16,16,X,Y+8,MODE);  X:=X+80;
    WSTAT(X,Y+12,'e');
    WSTAT(X+20,Y+12,'e');  Y:=Y-60;
  END;  (* FOR *)
END;  (* NETMETAL *)

(*------------------------------------------------------------------*)
PROCEDURE NETHYDROGEN;
(*------------------------------------------------------------------*)
VAR X,Y:INTEGER;
BEGIN
 X:=1;  Y:=YMAX-10;
 INITTURTLE;
 WSTAT(X,Y,'Net reaction of hydronium ions:-');  Y:=Y-80;
 DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X,Y-40,MODE);
 DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X,Y+5,MODE);  X:=X+80;
 WSTAT(X,Y,'e');
 WSTAT(X+20,Y,'e');  X:=X+50;
 DRAWARROW(X,Y,30,7);  X:=X+50;
 DRAWH2(X,Y-8);  X:=X+50;
 DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X,Y-35,MODE);
 DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X,Y+15,MODE);
END;  (* NETHYDROGEN *)

(*------------------------------------------------------------------*)
PROCEDURE NETREACTION(METEL:IONTYPE);
(*------------------------------------------------------------------*)
VAR SYMBOL:STRING[2]; AMETL:STRING[11];  CH:CHAR; X,Y:INTEGER;
BEGIN
 CASE METEL OF
  MG: BEGIN
      INITION(MG,CATION);  SYMBOL:='Mg'; AMETL:='magnesium:-';
     END;
  NI: BEGIN
      INITION(NI,CATION);   SYMBOL:='Ni'; AMETL:='nickel:-';
     END;
  END;
  X:=1;  Y:=YMAX-10;
  INITTURTLE;
  WSTAT(X,Y,CONCAT('Reaction between acid and ',AMETL));  Y:=Y-100;
  DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
  WSTAT(X+8,Y+12,SYMBOL);
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X+64,Y-22,MODE);
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X+64,Y+22,MODE);
  DRAWBLOCK(CATION,2,0,0,16,16,X+144,Y+8,MODE);
  DRAWH2(X+195,Y+5);
```

```
      DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X+240,Y-16,MODE);
      DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X+240,Y+25,MODE);
      DRAWARROW(X+100,Y+16,30,7);
    END; (* NETREACTION *)

    (*------------------------------------------------------------------*)
    PROCEDURE SHOWSPECTATOR;
    (*------------------------------------------------------------------*)
    VAR X,Y:INTEGER;
      (*..............................................................*)
      PROCEDURE DRAWCHLORIDE(CLX,CLY:INTEGER);
      (*..............................................................*)
      BEGIN
        DRAWBLOCK(ATOM,4,0,0,32,32,CLX,CLY,MODE);
        WSTAT(CLX+18,CLY+22,'-');
        WSTAT(CLX+8,CLY+12,'Cl');
      END; (* DRAWCHLORIDE *)

    BEGIN
     X:=1; Y:=YMAX-110;
     WSTAT(X,YMIN+12,'If hydrochloric acid is used, then');
     WSTAT(X,YMIN+2,'chloride is a spectator ion.');
     DRAWCHLORIDE(X+62,Y+55);
     DRAWCHLORIDE(X+62,Y-55);
     DRAWCHLORIDE(X+140,Y+35);
     DRAWCHLORIDE(X+140,Y-30);
    END; (* SHOWSPECTATOR *)

    (*------------------------------------------------------------------*)
    PROCEDURE CHECKKEY;
    (*------------------------------------------------------------------*)
    VAR CH:CHAR;
    BEGIN
     GETKEY(CH,[SPACE,'Q']);
     IF QUIT THEN EXIT(CONCLUSION);
    END; (* CHECKKEY *)


  BEGIN (*CONCLUSION*)
   NETMETAL;
   CHECKKEY;
   NETHYDROGEN;
   CHECKKEY;
   FOR METEL:=MG TO NI DO
    BEGIN
      NETREACTION(METEL);
      CHECKKEY;
      SHOWSPECTATOR;
      CHECKKEY;
    END;
  END; (* CONCLUSION *)
```

```
BEGIN  (* MICRO *)
  INITSHAPES;
  EXPLAINSHAPES;
   REPEAT
     IF NOT QUIT THEN REACTION;
     IF NOT QUIT THEN CONCLUSION;
     IF NOT QUIT THEN SOLVATECATIONS;
     PAGE(OUTPUT);
   UNTIL FIN('MICRO');
END; (* MICRO *)



(* METAL6 *)
(*************************************************************)
PROCEDURE MACRO;
(*************************************************************)
TYPE
  BITSHAPE=PACKED ARRAY[0..5,0..15]OF BOOLEAN;
  BUBSHAPE=PACKED ARRAY[0..5,0..7]OF BOOLEAN;
  MEDSIZE=PACKED ARRAY[0..7,0..7]OF BOOLEAN;
VAR
  BUBBLE:BUBSHAPE;                          (* shape of bubbles        *)
  FLAME:MEDSIZE;                            (* shape of flame          *)
  METAL:  ARRAY [1..3] OF BITSHAPE;         (* 3 different metal shapes *)

    (*=========================================================*)
    PROCEDURE INITSHAPES;
    (*=========================================================*)
    VAR COL:INTEGER;

      (*-------------------------------------------------------*)
      PROCEDURE INITMETAL(ROW:INTEGER;VAR BITS:BITSHAPE;S:STRING);
      (*-------------------------------------------------------*)
      BEGIN
        FOR COL:=0 TO 15 DO   BITS[ROW,COL]:=S[COL+1]='X';
      END; (* INITMETAL *)

      (*-------------------------------------------------------*)
      PROCEDURE INITBUBBLE(ROW:INTEGER;VAR BITS:BUBSHAPE;S:STRING);
      (*-------------------------------------------------------*)
      BEGIN
        FOR COL:=0 TO 7 DO  BITS[ROW,COL]:=S[COL+1]='X';
      END; (* INITBUBBLE *)

      (*-------------------------------------------------------*)
      PROCEDURE INITFLAME(ROW:INTEGER;VAR BITS:MEDSIZE;S:STRING);
      (*-------------------------------------------------------*)
      BEGIN
        FOR COL:=0 TO 7 DO  BITS[ROW,COL]:=S[COL+1]='X';
      END; (* INITFLAME *)
```

```
BEGIN (* INITSHAPES *)
  INITMETAL(5,METAL[1],' XXX   XXXXXX   ');
  INITMETAL(4,METAL[1],' X XX XXX    XXX');
  INITMETAL(3,METAL[1],'X    X         X');
  INITMETAL(2,METAL[1],'X              X');
  INITMETAL(1,METAL[1],'X      XXXX    X');
  INITMETAL(0,METAL[1],' XXXXXXXX  XXXX ');

  INITMETAL(5,METAL[2],'                ');
  INITMETAL(4,METAL[2],'  XX      XXX   ');
  INITMETAL(3,METAL[2],' X  X XX   X    ');
  INITMETAL(2,METAL[2],'   X XX    X    ');
  INITMETAL(1,METAL[2],'    X  XXXX     ');
  INITMETAL(0,METAL[2],'    XXXX        ');

  INITMETAL(5,METAL[3],'                ');
  INITMETAL(4,METAL[3],'                ');
  INITMETAL(3,METAL[3],'    X    X      ');
  INITMETAL(2,METAL[3],'   X X XX X     ');
  INITMETAL(1,METAL[3],'    X X XX      ');
  INITMETAL(0,METAL[3],'     XXX        ');
  INITBUBBLE(5,BUBBLE,'   XX   ');
  INITBUBBLE(4,BUBBLE,'  X  X  ');
  INITBUBBLE(3,BUBBLE,' X    X ');
  INITBUBBLE(2,BUBBLE,' X    X ');
  INITBUBBLE(1,BUBBLE,'  X  X  ');
  INITBUBBLE(0,BUBBLE,'   XX   ');

  INITFLAME(7,FLAME,'   XX   ');
  INITFLAME(6,FLAME,'   XX   ');
  INITFLAME(5,FLAME,'  X  X  ');
  INITFLAME(4,FLAME,'  X  X  ');
  INITFLAME(3,FLAME,' X    X ');
  INITFLAME(2,FLAME,' X    X ');
  INITFLAME(1,FLAME,'  X  X  ');
  INITFLAME(0,FLAME,'   XX   ');
END; (* INITVARS *)

(*================================================================*)
PROCEDURE REACTMETAL;
(*================================================================*)
CONST
 TTUBEX=120; (*x-coord at bottom centre of test-tube *)
 TTUBEY=36;  (*y-coord at bottom centre of test-tube *)
 TTWIDTH=32; (*width of test tube *)
 TTSIZE=116; (* height of test tube*)
 TTLEVEL=112; (*y-coord of level of soln in test tube *)
 METALX=128; (*x-coord for drawing metal *)

TYPE  METTYPE=(MAGNES,NICKEL);
VAR  MET:METTYPE;              (* metal being tested *)
     SPACEPR : BOOLEAN;          (* flag for space bar *)
     BUBLX,BUBLY,
     METALY,            (* current yco-ord of metal  *)
     DY,GAP:INTEGER;
```

```
(*-------------------------------------------------------------------------*)
PROCEDURE INIT;
(*-------------------------------------------------------------------------*)
VAR SKIP:INTEGER;
BEGIN
 METALY:=182;
 BUBLX:=132; (*x & y coord. of bubbles depends on position*)
 BUBLY:=TTUBEY+2;        (* of test tube*)
 DY:=4; (* required for MOVEBUBBLES *)
 CASE MET OF
   MAGNES:SKIP:=3;
   NICKEL:SKIP:=5;
   END;(*CASE*)
 GAP:=SKIP*DY;
END;(* INIT *)

(*-------------------------------------------------------------------------*)
PROCEDURE DRAWTTUBE(TUBEX,TUBEY,WIDTH,SIZE,LEVEL:INTEGER;
                    COL: SCREENCOLOR);
(*-------------------------------------------------------------------------*)
VAR EIGHTH,SIXNTH,RWIDTH:REAL;  X,Y:ARRAY[1..15]OF INTEGER;
     J: INTEGER;
BEGIN
 RWIDTH:=WIDTH;
 EIGHTH:=RWIDTH/8;
 Y[8]:=ROUND(EIGHTH*1.5);
 Y[4]:=ROUND(EIGHTH*1.25);
 Y[12]:=Y[4];
 Y[2]:=ROUND(EIGHTH*0.75);
 Y[14]:=Y[2];
 Y[1]:=ROUND(EIGHTH*0.5);
 Y[15]:=Y[1];
 Y[3]:=(Y[2]+Y[4])DIV 2;
 Y[13]:=Y[3];
 Y[6]:=ROUND(EIGHTH*1.4);
 Y[10]:=Y[6];
 Y[5]:=ROUND(EIGHTH*1.32);
 Y[11]:=Y[5];
 Y[7]:=Y[8];
 Y[9]:=Y[8];

 SIXNTH:=RWIDTH/16;
 FOR J:=1 TO 15 DO  X[J]:=ROUND(SIXNTH*J);

 MOVECOL(TUBEX,TUBEY+SIZE,COL);
 MOVETO(TUBEX,TUBEY);
 FOR J:=1 TO 15 DO MOVETO(TUBEX+X[J],TUBEY-Y[J]);
 MOVETO(TUBEX+WIDTH,TUBEY);
 MOVETO(TUBEX+WIDTH,TUBEY+SIZE);
 MOVETO(TUBEX+WIDTH,LEVEL);
 MOVECOL(TUBEX,LEVEL,NONE);
 WSTAT(TTUBEX+TTWIDTH+10,TTUBEY+40,'dilute');
 WSTAT(TTUBEX+TTWIDTH+18,TTUBEY+30,'HCl');
END; (* DRAWTTUBE *)
```

```
(*---------------------------------------------------------------------*)
PROCEDURE CHECKKEY(VAR SP :BOOLEAN);
(*---------------------------------------------------------------------*)
VAR CH :CHAR;
BEGIN
  READ(CH);
  QUIT :=CH='Q';
  SP :=CH=SPACE;
END; (* CHECKKEY *)

(*---------------------------------------------------------------------*)
PROCEDURE DRAWBUBBLES(X,Y :INTEGER);
(*---------------------------------------------------------------------*)
   (*.................................................................*)
   PROCEDURE EXTRABUBBLES(X,NEWY :INTEGER);
   (*.................................................................*)
   (* draw a column of bubbles vertically underneath  first bubble until at bottom of
      test tube TTUBEY*)
   VAR MORE :BOOLEAN;
   BEGIN
     REPEAT
       NEWY :=NEWY-GAP;
       MORE :=NEWY>=TTUBEY;
       IF MORE  THEN DRAWBLOCK(BUBBLE,2,0,0,8,6,X,NEWY,MODE)
     UNTIL NOT MORE;
   END; (* EXTRABUBBLES *)

BEGIN (* DRAWBUBBLES *)
  DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE); (*draw one bubble*)
  EXTRABUBBLES(X,Y);
END; (* DRAWBUBBLES *)

(*---------------------------------------------------------------------*)
PROCEDURE MOVEBUBBLES(VAR X,TOPY : INTEGER);
(*---------------------------------------------------------------------*)
VAR Y :INTEGER;
BEGIN (* MOVEBUBBLES *)
  Y :=TOPY;
  TOPY :=TOPY+DY;
  IF TOPY>TTLEVEL THEN TOPY :=TOPY-GAP;
  REPEAT
    DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE); (*erase *)
    IF (Y+DY)<TTLEVEL THEN
     DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE); (*display*)
    Y :=Y-GAP;
  UNTIL Y<TTUBEY;
  IF (Y+DY)>=TTUBEY THEN  DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE);
                                          (*display new bubble at bottom*)
END; (* MOVEBUBBLES *)
```

```
(*-------------------------------------------------------------------*)
PROCEDURE STATEMENT(AMET:METTYPE);
(*-------------------------------------------------------------------*)
VAR S1,S2:STRING;
BEGIN
 CASE AMET OF
  MAGNES:S1 :='Magnesium';
  NICKEL: S1 :='Nickel';
  END; (*CASE*)
 S1 :=CONCAT(S1,' dissolves in HCl solution');
 S2:='and a gas is evolved';
 WSTAT(20,13,S1);
 WSTAT(40,1,S2);
END; (* STATEMENT *)

(*-------------------------------------------------------------------*)
PROCEDURE RESULT(AMET:METTYPE);
(*-------------------------------------------------------------------*)
VAR CH:CHAR;
  (*.................................................................*)
  PROCEDURE HYDROG;
  (*.................................................................*)
  VAR S1,S2:STRING;.
  BEGIN
    S1 :='Gas produced was explosive';
    S2:='this indicates hydrogen gas';
    WSTAT(30,13,S1);
    WSTAT(40,1,S2);
  END; (*HYDROG*)

BEGIN (*RESULT*)
 DRAWBUBBLES(BUBLX,BUBLY);   (*erase bubbles *)
 STATEMENT(AMET); (* erase info that metal dissolves*)
 HYDROG;
 GETKEY(CH,[SPACE,'Q']);
 HYDROG;
END;  (* RESULT *)

(*-------------------------------------------------------------------*)
PROCEDURE DROPNDISSOLVE(AMET:METTYPE);
(*-------------------------------------------------------------------*)
  (*.................................................................*)
  PROCEDURE DROPMETAL(VAR Y:INTEGER;BOTTOM:INTEGER;AMETAL:METTYPE);
  (*.................................................................*)
  VAR CH:CHAR;

    PROCEDURE SPACEPROMPT;
    VAR S:STRING;
    BEGIN
     CASE AMETAL OF
        MAGNES:S:='MAGNESIUM';
        NICKEL:S:='NICKEL';
      END; (*CASE*)
     WSTAT(160,Y,S);
     WSTAT(30,5,'Press <SPACE BAR> to add metal');
    END; (*SPACEPROMPT*)
```

```
  BEGIN (* DROPMETAL *)
    DRAWBLOCK(METAL[1],2,0,0,16,6,METALX,Y,MODE);
    SPACEPROMPT;                    (*display*)
    GETKEY(CH,[SPACE,'Q']);
    SPACEPROMPT;
    IF QUIT THEN EXIT(REACTMETAL);
    REPEAT
      DRAWBLOCK(METAL[1],2,0,0,16,6,METALX,Y,MODE);
      Y:=Y-10;                  (* erase *)
      DRAWBLOCK(METAL[1],2,0,0,16,6,METALX,Y,MODE);
      DELAY(20);                 (*display*)
    UNTIL Y<=(BOTTOM+4);
  END; (* DROPMETAL *)

  (*..............................................................................*)
  PROCEDURE REQUEST;
  (*..............................................................................*)
  BEGIN
   WSTAT(30,183,'Press <SPACE BAR> to test gas');
  END; (* REQUEST *)


BEGIN (* DROPNDISSOLVE *)
  DROPMETAL(METALY,TTUBEY,MET);
  SPACEPR:=FALSE;
  STATEMENT(AMET);(*inform that metal dissoves & gas formed*)
  DRAWBUBBLES(BUBLX,BUBLY);  (*display bubbles*)
  REQUEST;    (*prompt for space bar to test gas*)
  REPEAT
   REPEAT
     MOVEBUBBLES(BUBLX,BUBLY);
   UNTIL KEYIN;
   CHECKKEY(SPACEPR);
   IF QUIT THEN EXIT(REACTMETAL);
  UNTIL SPACEPR;
  REQUEST;     (* erase prompt for space bar *)
  SPACEPR:=FALSE;
END; (* DROPNDISSOLVE *)

(*------------------------------------------------------------------------*)
PROCEDURE TESTGAS;
(*------------------------------------------------------------------------*)
VAR MATCHX,MATCHY,BUBBLES: INTEGER;

  (*..............................................................................*)
  PROCEDURE DRAWMATCH(X,Y:INTEGER;COL:SCREENCOLOR);
  (*..............................................................................*)
  BEGIN
   MOVECOL(X-12,Y+8,COL);
   MOVETO(X,Y);
   MOVECOL(X,Y+4,NONE);
   DRAWBLOCK(FLAME,2,0,0,8,8,X-3,Y+6,MODE);
  END; (* DRAWMATCH *)
```

```
(*..........................................................................................*)
PROCEDURE MOVEMATCH(VAR MATCHX,MATCHY:INTEGER);
(*..........................................................................................*)
CONST DX=8; DY=3;
BEGIN
 DRAWMATCH(MATCHX,MATCHY,BLACK2);
 MATCHX:=MATCHX+DX;
 MATCHY:=MATCHY-DY;
 DRAWMATCH(MATCHX,MATCHY,WHITE2);
END; (* MOVEMATCH *)

(*..........................................................................................*)
PROCEDURE EXPLOSION(CX,CY:INTEGER;COL:SCREENCOLOR);
(*..........................................................................................*)
CONST  DIST=12;

  PROCEDURE SHOWSLASH(ANGLE: INTEGER);
  BEGIN
   MOVETO(CX,CY);
   TURNTO(ANGLE);
   MOVE(DIST);
   PENCOLOR(COL);
   MOVE(DIST);
   PENCOLOR(NONE);
  END; (* SHOWSLASH *)

BEGIN (* EXPLOSION *)
 SHOWSLASH(0);
 SHOWSLASH(45);
 SHOWSLASH(90);
 SHOWSLASH(135);
 SHOWSLASH(180);
 SHOWSLASH(225);
 SHOWSLASH(315);
END; (* EXPLOSION *)

BEGIN (* TESTGAS *)
 MATCHX:=62;MATCHY:=180;
 DRAWMATCH(MATCHX,MATCHY,WHITE2); (* draw match *)
 REPEAT
   MOVEBUBBLES(BUBLX,BUBLY);
   MOVEMATCH(MATCHX,MATCHY);
 UNTIL MATCHX>=BUBLX;
 DRAWMATCH(MATCHX,MATCHY,BLACK2); (*erase match*)
 EXPLOSION(MATCHX,MATCHY+20,WHITE2); (* draw explosion*)

 FOR BUBBLES:=1 TO 4 DO MOVEBUBBLES(BUBLX,BUBLY);
 EXPLOSION(MATCHX,MATCHY+20,BLACK2); (*erase explosion*)
 IF KEYIN THEN CHECKKEY(SPACEPR);
 IF QUIT THEN EXIT(REACTMETAL);
END; (* TESTGAS *)

(*------------------------------------------------------------------------*)
PROCEDURE KEEPDISSOLVING(AMET:METTYPE);
(*------------------------------------------------------------------------*)
```

```
(*...........................................................................*)
PROCEDURE CYCLE(NUMCYCLES:INTEGER);
(*...........................................................................*)
VAR CYCLES: INTEGER;
BEGIN
 CYCLES:=0;
 REPEAT
   MOVEBUBBLES(BUBLX,BUBLY);
   IF KEYIN THEN CHECKKEY(SPACEPR);
   CYCLES:=CYCLES+1;
 UNTIL ((CYCLES>=NUMCYCLES) OR QUIT OR SPACEPR);
 IF QUIT THEN EXIT(REACTMETAL);
 IF SPACEPR THEN EXIT(KEEPDISSOLVING);
END; (* CYCLE *)

(*...........................................................................*)
PROCEDURE SWAPMETAL(OLD,NEW:INTEGER);
(*...........................................................................*)
(*erase old metal shape & display smaller shape   *)
BEGIN
 DRAWBLOCK(METAL[OLD],2,0,0,16,6,METALX,METALY,MODE);
 DRAWBLOCK(METAL[NEW],2,0,0,16,6,METALX,METALY,MODE);
END; (* SWAPMETAL *)

BEGIN
  CYCLE(12);
  SWAPMETAL(1,2);
  CYCLE(20);
  SWAPMETAL(2,3);
  CYCLE(20);
  DRAWBLOCK(METAL[3],2,0,0,16,6,METALX,METALY,MODE);
END; (* KEEPDISSOLVING *)            (*erase*)


BEGIN (* REACTMETAL *)
 FOR MET:=MAGNES TO NICKEL DO
  BEGIN
    INITTURTLE;
    INIT;
    DRAWTTUBE(TTUBEX,TTUBEY,TTWIDTH,TTSIZE,TTLEVEL,WHITE1);
    DROPNDISSOLVE(MET);
    TESTGAS;
    KEEPDISSOLVING(MET);
    IF NOT QUIT THEN RESULT(MET);
  END;
END; (* REACTMETAL *)
```

```
(*METAL7*)
  (*==================================================================*)
  PROCEDURE CONCLUSION;
  (*==================================================================*)
  CONST WIDTH=8;

    (*----------------------------------------------------------------*)
    PROCEDURE BIGSTAT(Y:INTEGER; S:STRING);
    (*----------------------------------------------------------------*)
    VAR LETTER,NUM,X :INTEGER; CH:CHAR;
    BEGIN
     X:=(XMAX-(10*LENGTH(S))) DIV 2;
     FOR LETTER:=1 TO LENGTH(S) DO
      BEGIN
       MOVETO(X,Y);
       NUM:=ORD(S[LETTER]);
       IF NUM IN [65..90] THEN NUM:=NUM-65  (* A..Z *)
        ELSE IF NUM=43 THEN NUM:=26; (* + *)
       WCHAR(CHR(NUM));
       X:=X+10;
      END;
    END; (* BIGSTAT *)

    (*----------------------------------------------------------------*)
    PROCEDURE ENCLOSE(WIDTH:INTEGER; COL:SCREENCOLOR);
    (*----------------------------------------------------------------*)
    BEGIN
     FILLAREA(XMIN,XMIN+WIDTH,YMIN,YMAX,COL);
     FILLAREA(XMAX-WIDTH,XMAX,YMIN,YMAX,COL);
     FILLAREA(XMIN,XMAX,YMIN,YMIN+WIDTH,COL);
     FILLAREA(XMIN,XMAX,YMAX-WIDTH,YMAX,COL);
    END; (* ENCLOSE *)

    (*----------------------------------------------------------------*)
    PROCEDURE DOWNARROW(X,Y1,Y2:INTEGER; COL:SCREENCOLOR);
    (*----------------------------------------------------------------*)
    CONST WID=3;
    VAR X1,X2:INTEGER;
    BEGIN
     FILLAREA(X-WID,X+WID,Y2,Y1,COL);
     X1:=X-3*WID;
     X2:=X+3*WID;
     REPEAT
      DRAWLINE(X1,Y2,X2,Y2,COL);
      Y2:=Y2-1;
      X1:=X1+1;
      X2:=X2-1;
     UNTIL X1>=X2;
    END; (* DOWNARROW *)

    (*----------------------------------------------------------------*)
    PROCEDURE TEXT;
    (*----------------------------------------------------------------*)
    TYPE METAL=(MG,AL,NI,PB);   ANACID=(CHLORIC,SULF);
    VAR  BAND:INTEGER;
         AMETAL:METAL;  ANYACID:ANACID;
```

```
(*..............................................................................................*)
PROCEDURE GENERAL;
(*..............................................................................................*)
VAR Y:INTEGER; CH:CHAR;
BEGIN
 Y:=YMAX-40;
 BIGSTAT(Y,'ACID + ACTIVE METAL');
 Y:=Y-25;
 DOWNARROW(XMAX DIV 2,Y,Y-25,WHITE);
 Y:=Y-75;
 BIGSTAT(Y,'HYDROGEN + SALT');
 GETKEY(CH,[SPACE,'Q']);
 IF QUIT THEN EXIT(TEXT);
END; (* GENERAL *)

(*..............................................................................................*)
PROCEDURE EQUATION(ACID:ANACID);
(*..............................................................................................*)
VAR Y:INTEGER; CH:CHAR;
   S1,S2,METSTR:STRING;
BEGIN
 METSTR:='MAGNESIUM';
 CASE ACID OF
     CHLORIC:BEGIN S1:='HYDROCHLORIC ACID';  S2:=' CHLORIDE'; END;
     SULF   :BEGIN S1:='SULFURIC ACID'; S2:=' SULFATE'; END;
   END; (*CASE*)
 Y:=YMAX-30;
 BIGSTAT(Y,S1); Y:=Y-15;
 BIGSTAT(Y,'+'); Y:=Y-15;
 BIGSTAT(Y,METSTR); Y:=Y-15;
 DOWNARROW(XMAX DIV 2,Y,Y-20,WHITE); Y:=Y-55;
 BIGSTAT(Y,'HYDROGEN'); Y:=Y-15;
 BIGSTAT(Y,'+'); Y:=Y-15;
 BIGSTAT(Y,CONCAT(METSTR,S2));
 GETKEY(CH,[SPACE,'Q']);
 IF QUIT THEN EXIT(TEXT);
END; (* EQUATION *)

(*..............................................................................................*)
PROCEDURE CHANGE(MET:METAL; ACID:ANACID);
(*..............................................................................................*)
VAR Y:INTEGER; CH:CHAR;  SALT:STRING[10];  METSTR:STRING[10];

   PROCEDURE BLANKLINE(Y,X1,X2:INTEGER);
   CONST BLANK='          ';
   BEGIN
    WHILE X1<X2 DO  BEGIN  WSTAT(X1,Y,BLANK); X1:=X1+70  END;
   END; (* BLANKLINE *)

BEGIN (* CHANGE *)
 CASE ACID OF
     CHLORIC  : SALT:=' CHLORIDE';
     SULF     : SALT:=' SULFATE';
   END; (*CASE*)
```

```
          CASE MET OF
            AL: METSTR:='ALUMINIUM';
            PB: METSTR:='LEAD';
            NI: METSTR:='NICKEL';
           END; (*CASE*)
          Y:=YMAX-60;
          BLANKLINE(Y,XMIN+20,XMAX-60);
          BIGSTAT(Y,METSTR);
          Y:=Y-100;
          BLANKLINE(Y,XMIN+20,XMAX-60);
          BIGSTAT(Y,CONCAT(METSTR,SALT));
          GETKEY(CH,[SPACE,'Q']);
          IF QUIT THEN EXIT(TEXT);
         END; (* CHANGE *)

      BEGIN (*TEXT*)
       GENERAL;
       BAND:=WIDTH+5;
       FOR ANYACID:=CHLORIC TO SULF DO
        BEGIN
         FILLAREA(XMIN+BAND,XMAX-BAND,YMIN+BAND,YMAX-BAND,BLACK);
         EQUATION(ANYACID);
         FOR AMETAL:=AL TO PB DO CHANGE(AMETAL,ANYACID);
        END;
      END; (* TEXT *)


    BEGIN (* CONCLUSION *)
     CHARTYPE(10);
     INITTURTLE;
     ENCLOSE(WIDTH,VIOLET);
     TEXT;
     CHARTYPE(6)
    END; (* CONCLUSION *)

BEGIN (* MACRO *)
  INITSHAPES;
  REPEAT
     REACTMETAL;
     IF NOT QUIT THEN CONCLUSION;
  UNTIL FIN('MACRO');
  TEXTMODE;
END; (* MACRO *)
```

```
(****************************************************************)
PROCEDURE SELECT(VAR CH:CHAR);
(****************************************************************)
CONST DOTS='......(';      DEMO='SCOPIC demonstration    ';
VAR  X,Y: INTEGER;
BEGIN
 TEXTMODE;
 PAGE(OUTPUT);
 X:=0; Y:=1;
 WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
 WRITE(AT(4,Y),'REACTION BETWEEN AN ACTIVE METAL'); Y:=Y+1;
 WRITE(AT(14,Y),'AND AN ACID'); Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+4;
 WRITE(AT(X,Y),'MACRO',DEMO,DOTS,'1)'); Y:=Y+3;
 WRITE(AT(X,Y),'MICRO',DEMO,DOTS,'2)'); Y:=Y+3;
 WRITE(AT(X,Y),'QUIT - back to main menu     ',DOTS,'Q)'); Y:=Y+4;
 WRITE(AT(X+10,Y),'Select option ....',DOTS,'  )');
 GETTEXTCHAR(37,Y,CH,['1','2','Q']);
 QUIT:=CH='Q';
 PAGE(OUTPUT);
END; (* SELECT *)

(****************************************************************)
PROCEDURE BACKTOMENU;
(****************************************************************)
BEGIN
  SETCHAIN(' :DEMOMENU');
  PAGE(OUTPUT);
  WRITE(AT(10,8),'R E L O A D I N G');
  WRITE(AT(10,11),'M A I N  M E N U ........');
END; (* BACKTOMENU *)

BEGIN  (* MAIN *)
 FALSEARRAY(BLANK);
 INITACID; (* required in MICRO but used globally as *)
 INITMG;   (* they take a while to initialise *)
 CHARTYPE(MODE);
 REPEAT
  SELECT(OPTION);
  IF NOT QUIT THEN
   BEGIN
   CASE OPTION OF
     '1' : MACRO;
     '2' : MICRO;
     END; (*CASE*)
    QUIT:=FALSE;
   END;
  UNTIL QUIT;
  BACKTOMENU;
END. (*METAL*)
```

```
(*$S++*)
(* Macroscopic & microscopic demonstration of reaction between a very reactive metal
and water *)

PROGRAM ACTIVEMETAL;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;

CONST  MODE=6;
TYPE
 LARGSIZ=PACKED ARRAY[1..24,1..32] OF BOOLEAN;
 MOLECULE= RECORD
             X,Y,DX,DY : INTEGER;
             SHAPE :LARGSIZ;
           END;
 PH=(NEUTRAL,ACIDIC,BASIC);
 BIGSHAPE=PACKED ARRAY[1..32,1..32] OF BOOLEAN;

VAR
  ATOM: BIGSHAPE;                           (* shape of metal atoms   *)
  BLANK :LARGSIZ;
  WATERMOL,                                 (*shape & position of H20 *)
  HYDROXIDE :ARRAY[1..2] OF MOLECULE;       (*shape & position of OH- *)
  QUIT : BOOLEAN;
  OPTION: CHAR;

(**********************************************************)
PROCEDURE GETKEY(VAR ACH:CHAR; LEGALSET :CHARSET);
(**********************************************************)
BEGIN
 GETACHAR(ACH,LEGALSET);
 QUIT :=(ACH='Q');
END; (*GETKEY*)


(**********************************************************)
PROCEDURE FALSEARRAY(VAR NEWARRAY :LARGSIZ);
(**********************************************************)
VAR ROW,COL : INTEGER;
BEGIN
 FOR ROW :=1 TO 24 DO
   FOR COL :=1 TO 32 DO  NEWARRAY[ROW,COL]:=FALSE;
END; (* FALSEARRAY*)

(**********************************************************)
PROCEDURE  DEFINESHAPE(VAR NEWARRAY :LARGSIZ; ACIDITY :PH; ANUM:INTEGER);
(**********************************************************)
(* Defines shape of water & hydroxide molecules. 2 orientations (1 & 2 determin by
'ANUM') of each molecule is availabe.The array , 'NEWARRAY' must first be initialized to
all false!*)
VAR MAXCOL, HEIGHT, WIDTH : INTEGER;

  (*=====================================================*)
  PROCEDURE INIT(WIDTH,HEIGHT :INTEGER; SYMBOL :CHAR);
  (*=====================================================*)
  VAR MAXCOL : INTEGER;
```

```
(*-----------------------------------------------------------------*)
PROCEDURE MERGE(ROW:INTEGER; S:STRING);
(*-----------------------------------------------------------------*)
VAR COL: INTEGER;
BEGIN
 FOR COL:=1 TO MAXCOL DO
            NEWARRAY[ROW+HEIGHT,COL+WIDTH]:= S[COL]='X';
END; (* MERGE *)

BEGIN (*INIT*)
 CASE SYMBOL OF
 'H': BEGIN
     MAXCOL:=5;
     MERGE(5,'X   X');
     MERGE(4,'X   X');
     MERGE(3,'XXXXX');
     MERGE(2,'X   X');
     MERGE(1,'X   X');
   END; (* INITH *)
 'O': BEGIN
     MAXCOL:=9;
     MERGE(8,'  XXXXX  ');
     MERGE(7,' X     X ');
     MERGE(6,'X       X');
     MERGE(5,'X       X');
     MERGE(4,'X       X');
     MERGE(3,'X       X');
     MERGE(2,' X     X ');
     MERGE(1,'  XXXXX  ');
   END; (* INITO *)
 '1': BEGIN
     MAXCOL:=4;
     MERGE(1,'XXXX');
   END; (* INIT1BOND *)
 '2': BEGIN
     MAXCOL:=3;
     MERGE(3,'  X');
     MERGE(2,' X ');
     MERGE(1,'X  ');
   END; (* INIT2BOND *)
 '3':BEGIN
     MAXCOL:=3;
     MERGE(3,'X  ');
     MERGE(2,' X ');
     MERGE(1,'  X');
   END; (* INIT3BOND *)
 END; (*CASE*)
END; (* INIT *)

BEGIN (* DEFINESHAPE *)
 CASE ACIDITY OF
 NEUTRAL: CASE ANUM OF
     1 : BEGIN
        INIT(12,10,'O');
        INIT(27,0,'H'); (* removed in reaction *)
        INIT(0,11,'H');
        INIT(6,13,'1');
        INIT(22,6,'3'); (* broken in reaction *)
```

```
            INIT(6,13,'1');
            INIT(22,6,'3'); (* broken in reaction *)
          END;
        2 :BEGIN
            INIT(9,10,'0');
            INIT(0,0,'H'); (* donated in reaction *)
            INIT(26,11,'H');
            INIT(21,13,'1'); INIT(6,6,'2'); (*bond broken*)
          END;
        3: BEGIN
           INIT(14,8,'0');
           INIT(0,0,'H');
           INIT(0,16,'H');
           INIT(8,4,'2');
           INIT(8,15,'3');
          END;
        4:  BEGIN
            INIT(0,8,'0');
            INIT(16,0,'H');
            INIT(16,16,'H');
            INIT(10,15,'2');
            INIT(10,4,'3');
          END;
        5:BEGIN
            INIT(8,0,'0');
            INIT(0,16,'H');
            INIT(18,16,'H');
            INIT(14,10,'2');
            INIT(7,10,'3');
          END;
        6:BEGIN
            INIT(8,16,'0');
            INIT(0,3,'H');
            INIT(16,3,'H');
            INIT(5,9,'2');
            INIT(14,9,'3');
          END;
        END; (*CASE*)
   BASIC : CASE ANUM OF
       1 : BEGIN
           (*all y ordinates will be 10 less than
             in corresponding neutral shape *)
           INIT(12,0,'0');
           INIT(0,1,'H');
           INIT(6,3,'1');
           INIT(8,10,'1'); (* neg. charge*)
          END;
       2: BEGIN
           INIT(9,0,'0');
           INIT(26,1,'H');
           INIT(21,3,'1');
           INIT(20,10,'1'); (* neg. charge *)
          END;
       END;(*CASE*)
    END; (*CASE*)
END; (*DEFINESHAPE*)
```

```
(***********************************************************)
PROCEDURE INITWATER;
(***********************************************************)
VAR J: INTEGER;
BEGIN
  FOR J:=1 TO 2 DO
   BEGIN
    WITH WATERMOL[J] DO
     BEGIN
       SHAPE:=BLANK;
       DEFINESHAPE(SHAPE,NEUTRAL,J);
     END; (*WITH*)
    WITH HYDROXIDE[J] DO
     BEGIN
       SHAPE:=BLANK;
       DEFINESHAPE(SHAPE,BASIC,J);
     END; (* WITH *)
    END; (* FOR *)
 END; (* INITWATER *)

(***********************************************************)
PROCEDURE INITMETAL;
(***********************************************************)
CONST MAXCOL=32;
VAR  STR: ARRAY[1..MAXCOL] OF STRING; J:INTEGER;

  (*=========================================================*)
  PROCEDURE INIT(ROW:INTEGER; VAR BITS:BIGSHAPE; S:STRING);
  (*=========================================================*)
  VAR  COL: INTEGER;
  BEGIN
   FOR COL:=1 TO MAXCOL DO  BITS[ROW,COL]:=S[COL]='X';
  END;  (* INIT*)

BEGIN
   STR[1]:='           XXXXXXXX             ';
   STR[2]:='        XXXXXXXXXXXXXXX         ';
   STR[3]:='     XXXXXXXXXXXXXXXXXXX        ';
   STR[4]:='    XXXXXXXXXXXXXXXXXXXXX       ';
   STR[5]:='   XXXXXXXXXXXXXXXXXXXXXXX      ';
   STR[6]:='   XXXXXXXXXXXXXXXXXXXXXXXXX    ';
   STR[7]:='  XXXXXXXXXXXXXXXXXXXXXXXXXX    ';
   STR[8]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXX    ';
   STR[9]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXX   ';
  STR[10]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXX   ';
  STR[11]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
  STR[12]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
  STR[13]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
  STR[14]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
  STR[15]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
  STR[16]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
   FOR J:=1 TO 16 DO
   BEGIN
     INIT(J,ATOM,STR[J]);  INIT(33-J,ATOM,STR[J]);
   END;
END; (* INITMETAL *)
```

```
(**********************************************************)
FUNCTION FIN(STR:STRING):BOOLEAN;
(**********************************************************)
VAR CH:CHAR;
BEGIN
  INITTURTLE;
  WSTAT(30,100,CONCAT('Repeat ',STR,'SCOPIC'));
  WSTAT(30,70,'demonstration? (Y/N)');
  CHARTYPE(10);
  GETHICHAR(185,70,CH,['Y','N','Q']);
  CHARTYPE(MODE);
  FIN:=CH<>'Y';
  QUIT:=CH='Q';
END; (* FIN *)

(**********************************************************)
PROCEDURE MICRO;
(**********************************************************)
TYPE  MEDSHAPE=PACKED ARRAY[1..16,1..16] OF BOOLEAN;
      IONTYPE=(NA,CA,MET);
VAR METEL:IONTYPE;
    BLANKION,CATION:MEDSHAPE;
    HATOM:PACKED ARRAY[1..5,1..5]OF BOOLEAN;
    HYDROGEN:PACKED ARRAY[1..5,1..16]OF BOOLEAN;

   (*=====================================================*)
   PROCEDURE DRAWARROW(X,Y,SIZE: INTEGER);
   (*=====================================================*)
   CONST TIP=7;
   BEGIN
     MOVECOL(X,Y,WHITE1);
     MOVECOL(X+SIZE,Y,NONE);
     MOVECOL(X+SIZE-TIP,Y+TIP,WHITE1);
     MOVETO(X+SIZE,Y);
     MOVECOL(X+SIZE-TIP,Y-TIP,NONE);
   END; (*DRAWARROW*)

   (*=====================================================*)
   PROCEDURE INITION(METEL:IONTYPE; VAR ANYION:MEDSHAPE);
   (*=====================================================*)
       (*-------------------------------------------------*)
       PROCEDURE MERGEBITS(ROW:INTEGER;VAR BITS:MEDSHAPE; S:STRING);
       (*-------------------------------------------------*)
       VAR  COL:INTEGER;
       BEGIN
          FOR COL:=1 TO 16 DO BITS[ROW,COL]:=S[COL]='X';
       END; (* INITBITS *)

   BEGIN (*INITION *)
    ANYION:=BLANKION;
    CASE METEL OF
     NA :BEGIN
          MERGEBITS(10,ANYION,'X  XXX  XXXX');
           MERGEBITS(9,ANYION,'X   XX  XX     X');
           MERGEBITS(8,ANYION,'X  X X  X  XX  X');
           MERGEBITS(7,ANYION,'X  XX   X  XX  X');
           MERGEBITS(6,ANYION,'X  XXX  XX     X');
```

```
      END;
    CA :BEGIN
         MERGEBITS(15,ANYION,'   XXXXXXXXX    ');
         MERGEBITS(14,ANYION,'  XX  XXX  XXX  ');
         MERGEBITS(13,ANYION,' X     X     XX ');
         MERGEBITS(12,ANYION,' XXX  XXX  XXXX ');
         MERGEBITS(11,ANYION,'XXXXXXXXXXXXX');
         MERGEBITS(10,ANYION,'XX     XXXXXXXXX');
         MERGEBITS(9,ANYION,'X   XX  XXXXXXXX');
         MERGEBITS(8,ANYION,'X  XXXXXXX     X');
         MERGEBITS(7,ANYION,'X  XXX  X  XX  X');
         MERGEBITS(6,ANYION,'XX     XXX     X');
       END;
    MET:BEGIN
         MERGEBITS(9,ANYION,'XXXX   XX   XXXX');
         MERGEBITS(8,ANYION,'XXXX        XXXX');
         MERGEBITS(7,ANYION,'XXXX  X  X  XXXX');
         MERGEBITS(6,ANYION,'XXXX  X  X  XXXX');
         MERGEBITS(5,ANYION,' XXX  XXXX  XXX ');
       END;
    END;(*CASE*)
END;(*INITION*)

(*=============================================================*)
PROCEDURE INITSHAPES;
(*=============================================================*)
   (*-----------------------------------------------------------*)
   PROCEDURE INITBLANK(VAR ANYION:MEDSHAPE);
   (*-----------------------------------------------------------*)
      (*.......................................................*)
      PROCEDURE INITBITS(ROW:INTEGER;VAR BITS:MEDSHAPE; S:STRING);
      (*.......................................................*)
       VAR  COL:INTEGER;
       BEGIN
        FOR COL:=1 TO 16 DO BITS[ROW,COL]:=S[COL]='X';
       END; (* INITBITS *)

   BEGIN (*INITBLANK *)
    INITBITS(16,ANYION,'      XXXX      ');
    INITBITS(15,ANYION,'   XXXXX  XX    ');
    INITBITS(14,ANYION,'  XXXXXX  XXXX  ');
    INITBITS(13,ANYION,' XXXXX      XXX ');
    INITBITS(12,ANYION,' XXXXXXX  XXXXX ');
    INITBITS(11,ANYION,'XXXXXXXX  XXXXXX');
    INITBITS(10,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(9,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(8,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(7,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(6,ANYION,'XXXXXXXXXXXXXXXX');
    INITBITS(5,ANYION,' XXXXXXXXXXXXXX ');
    INITBITS(4,ANYION,'  XXXXXXXXXXXX  ');
    INITBITS(3,ANYION,'  XXXXXXXXXXXX  ');
    INITBITS(2,ANYION,'    XXXXXXXXX   ');
    INITBITS(1,ANYION,'     XXXXXX     ');
   END; (* INITBLANK*)
```

```
(*-----------------------------------------------------------------*)
PROCEDURE INITHATOM;
(*-----------------------------------------------------------------*)
  (*...............................................................*)
  PROCEDURE INIT(ROW:INTEGER; S:STRING);
  (*...............................................................*)
  VAR COL: INTEGER;
   BEGIN
    FOR COL:=1 TO 5 DO HATOM[ROW,COL]:=S[COL]='X';
   END; (* INIT *)

 BEGIN
  INIT(5,'X   X');
  INIT(4,'X   X');
  INIT(3,'XXXXX');
  INIT(2,'X   X');
  INIT(1,'X   X');
 END; (* INITHATOM *)

(*-----------------------------------------------------------------*)
PROCEDURE INITHYDROGEN;
(*-----------------------------------------------------------------*)

  PROCEDURE INIT(ROW:INTEGER; S:STRING);
  VAR COL:INTEGER;
  BEGIN
   FOR COL:=1 TO 16 DO HYDROGEN[ROW,COL]:=S[COL]='X';
  END; (* INIT *)

 BEGIN
  INIT(5,'X  X    X  X');
  INIT(4,'X  X    X  X');
  INIT(3,'XXXX X XXXX ');
  INIT(2,'X  X    X  X');
  INIT(1,'X  X    X  X');
 END; (* INITHYDROGEN*)

BEGIN
 INITBLANK(BLANKION);
 INITION(NA,CATION);
 INITHATOM;
 INITHYDROGEN;
END;(*INITSHAPES*)

(*$I :AMETAL2*)
(*$I :AMETAL3*)
(*$I :AMETAL4*)
(*$I :AMETAL5*)
(*$I :AMETAL6*)
(*$I :AMETAL7*)
(*$I :AMETAL8*)
```

```
(* AMETAL2 *)
  (*============================================================*)
  PROCEDURE EXPLAINSHAPES;
  (*============================================================*)

    (*----------------------------------------------------------*)
    PROCEDURE SHOWSHAPES;
    (*----------------------------------------------------------*)
    VAR CH:CHAR;
    BEGIN
      INITTURTLE;
      WSTAT(0,162,'    This demonstration will display');
      WSTAT(0,142,'following structures:-');
      DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,140,90,MODE);
      WSTAT(0,90,'WATER MOLECULE:');
      DRAWBLOCK(HYDROXIDE[1].SHAPE,4,0,0,32,16,140,20,MODE);
      WSTAT(0,20,'HYDROXIDE ION:');
      GETKEY(CH,[SPACE,'Q']);
      IF QUIT THEN EXIT(MICRO);
    END; (* SHOWSHAPES *)

    (*----------------------------------------------------------*)
    PROCEDURE ATOMSTRUCTURE;
    (*----------------------------------------------------------*)
    VAR METALX,METALY : INTEGER;
        CH: CHAR;

      (*........................................................*)
      PROCEDURE SHOWSODIUM(X,Y:INTEGER);
      (*........................................................*)
      BEGIN
       DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
       WSTAT(X,Y-12,'Sodium');
       WSTAT(X,Y-21,' Atom ');
       WSTAT(10,Y-70,'A sodium atom has 1 valence electron.');
      END; (* SHOWSODIUM *)
      (*........................................................*)
      PROCEDURE SHOWSTRUCTURE(X,Y:INTEGER);
      (*........................................................*)

        PROCEDURE ARROW(X,Y,SIZE:INTEGER);
        CONST TIP=3;
        BEGIN
          MOVECOL(X-SIZE+TIP,Y+TIP,WHITE2);
          MOVETO(X-SIZE,Y);
          MOVECOL(X-SIZE+TIP,Y-TIP,NONE);
          MOVECOL(X-SIZE+1,Y,WHITE2);
          MOVETO(X-SIZE,Y);
          MOVECOL(X,Y,NONE);
        END; (* ARROW *)

       BEGIN (* SHOWSTRUCTURE *)
        DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE);
        WSTAT(X+12,Y+26,'e');      (*display metal ion*)
        ARROW(X+40,Y+30,8);
        ARROW(X+40,Y+15,MODE);
```

```
      WSTAT(X+45,Y+26,'valence electron');
      WSTAT(X+45,Y+11,'positive ion');
    END; (* SHOWSTRUCTURE *)

BEGIN (* ATOMSTRUCTURE *)
  INITTURTLE;
  METALX:=110; METALY:=125;
  SHOWSODIUM(METALX,METALY);
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(MICRO);
  SHOWSTRUCTURE(METALX,METALY);
  GETKEY(CH,[SPACE,'Q']);
END; (* ATOMSTRUCTURE *)

(*---------------------------------------------------------------*)
PROCEDURE METALSTRUCTURE;
(*---------------------------------------------------------------*)
CONST  STARTX=80; STARTY=60;  NUMSTRS=4;
TYPE   MANYSTR=ARRAY[1..NUMSTRS] OF STRING;
VAR    S:MANYSTR;
   (*...........................................................*)
   PROCEDURE DRAWATOMS(X1,Y1:INTEGER);
   (*...........................................................*)
   CONST SIZE=32;
   TYPE OUTLINE=(OUTER,INNER);
   VAR  SHAPE:OUTLINE;
      X,Y:INTEGER; CH:CHAR;

      PROCEDURE DRAWAROW(XX,YY,ANYNUM:INTEGER);
      VAR  I: INTEGER;
      BEGIN
        FOR I:=1 TO ANYNUM DO
          BEGIN
           IF SHAPE=OUTER THEN  DRAWBLOCK(ATOM,4,0,0,32,32,XX,YY,MODE)
                  ELSE  DRAWBLOCK(CATION,2,0,0,16,16,XX+8,YY+8,MODE);
          YY:=YY+SIZE;
         END;
       END; (* DRAWAROW *)

   BEGIN  (* DRAWATOMS *)
     FOR SHAPE:=OUTER TO INNER DO
      BEGIN
        X:=X1; Y:=Y1;
        DRAWAROW(X,Y,2);
        X:=X+SIZE-5;
        DRAWAROW(X,Y-(SIZE DIV 2),3);
        X:=X+SIZE-5;
        DRAWAROW(X,Y,2);
        IF SHAPE=OUTER THEN GETKEY(CH,[SPACE,'Q']);
        IF QUIT THEN EXIT(METALSTRUCTURE);
      END;
   END; (* DRAWATOMS *)
```

```
(*..........................................................................................*)
PROCEDURE SHOWTEXT(VAR STR:MANYSTR);
(*..........................................................................................*)
CONST X=20;
VAR Y,J:INTEGER;
BEGIN
 Y:=182;
 FOR J:=1 TO NUMSTRS DO
  BEGIN
   WSTAT(X,Y,STR[J]);
   Y:=Y-12;
  END;
END; (* SHOWTEXT *)

(*..........................................................................................*)
PROCEDURE CRYSTAL(X1,Y1:INTEGER);
(*..........................................................................................*)
BEGIN
   S[1]:=' Sodium atoms are arranged in a';
   S[2]:='crystal structure in which each';
   S[3]:='sodium atom is surrounded by many';
   S[4]:='other sodium atoms. ';
   SHOWTEXT(S); (*display text*)
   DRAWATOMS(X1,Y1);
   FILLAREA(XMIN,XMAX,140,YMAX,BLACK1); (*erase*)
END;

(*..........................................................................................*)
PROCEDURE BONDING;
(*..........................................................................................*)
VAR CH:CHAR;
BEGIN
   S[1]:=' Metallic bonding is often';
   S[2]:=' described as positive';
   S[3]:=' metal ions embedded';
   S[4]:=' in an "electronic glue"';
   SHOWTEXT(S);
   GETKEY(CH,[SPACE,'Q']);
   FILLAREA(XMIN,XMAX,140,YMAX,BLACK1); (*erase*)
END; (* BONDING *)

(*..........................................................................................*)
PROCEDURE LOSELECTRONS(X1,Y1:INTEGER);
(*..........................................................................................*)
VAR ELECY,ELECX,K:INTEGER; CH:CHAR;

    PROCEDURE FLASHELECTRON(VAR X,Y:INTEGER; DX:INTEGER);
    BEGIN
      WSTAT(X,Y,'e'); (* erase electron*)
      X:=X-DX;
      DELAY(40);
      WSTAT(X,Y,'e'); (*display electron*)
      DELAY(120);
    END; (* FLASHELECTRON *)
```

```
            PROCEDURE REMOVEATOM(X,Y:INTEGER);
            VAR J: INTEGER;
            BEGIN
             DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,4);(*erase*)
             FOR J:=1 TO 7 DO
              BEGIN                 (*display*)
                DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE);
                DELAY(30);
                DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y+8,MODE);
                X:=X-8; Y:=Y-4;
              END; (* FOR*)             (*erase *)
             END; (* REMOVEATOM *)

        BEGIN (* LOSELECTRONS *)
          S[1]:='If an electron is removed from this ';
          S[2]:='crystal structure then a positive ion ';
          S[3]:='is lost from the crystal structure.';
          S[4]:=' ';
          SHOWTEXT(S);  (*display text*)
          ELECX:=X1+2; ELECY:=Y1+12;
          WSTAT(ELECX,ELECY,'e');  (* display electron*)
          S[1]:='Press <SPACE BAR> to remove electron';
          WSTAT(0,0,S[1]);  (* display *)
          CH:='X';
          REPEAT
             FLASHELECTRON(ELECX,ELECY,0);
             IF KEYIN THEN READ(CH);
          UNTIL (CH=SPACE);
          WSTAT(0,0,S[1]);  (* erase *)

          FOR K:=1 TO 3 DO FLASHELECTRON(ELECX,ELECY,4);
          WSTAT(ELECX,ELECY,'e');  (* erase electron *)
          REMOVEATOM(X1,Y1);
          GETKEY(CH,[SPACE,'Q']);
        END; (* LOSELECTRONS *)

    BEGIN (* METALSTRUCTURE *)
       INITTURTLE;
       CRYSTAL(STARTX,STARTY);
       BONDING;
       LOSELECTRONS(STARTX,STARTY);
    END; (* METALSTRUCTURE *)

BEGIN (* EXPLAINSHAPES *)
  SHOWSHAPES;
  ATOMSTRUCTURE;
  IF NOT QUIT THEN METALSTRUCTURE;
END; (* EXPLAINSHAPES *)
```

```
(* AMETAL3 *)
   (*============================================================*)
   PROCEDURE SOLVATECATIONS;
   (*============================================================*)
   CONST WATERNUM=4;
   TYPE  WATERSIZE=PACKED ARRAY[1..24,1..24]OF BOOLEAN;
   VAR  WATER: ARRAY[1..WATERNUM] OF WATERSIZE;
        AQUA :INTEGER;

     (*----------------------------------------------------------*)
     PROCEDURE DEFINEWATER;
     (*----------------------------------------------------------*)
     BEGIN  (* DEFINEWATER *)
      WHILE ((AQUA<=WATERNUM) AND (NOT KEYIN)) DO
       BEGIN
         WATER[AQUA]:=BLANK;
         DEFINESHAPE(WATER[AQUA],NEUTRAL,AQUA+2);
         AQUA:=AQUA+1;
       END;
     END; (* DEFINEWATER *)

     (*----------------------------------------------------------*)
     PROCEDURE HYDRATE(ANUM,CENTRX,CENTRY:INTEGER);
     (*----------------------------------------------------------*)
     VAR  SIZE,DISTANCE,RADIUS,BONDLEN,
          J,X,Y,X1,Y1,X2,Y2 :INTEGER;
     BEGIN
       SIZE:=24;
       DISTANCE:=20;
       BONDLEN:=4;
       RADIUS:=15;
        FOR J:=1 TO ANUM DO
         BEGIN
          CASE J OF
          1: BEGIN
               X:=CENTRX-DISTANCE-SIZE;
               Y:=CENTRY-(SIZE DIV 2);
               X1:=CENTRX-RADIUS-2; Y1:=CENTRY;
               X2:=X1+BONDLEN;  Y2:=Y1;
             END;
          2: BEGIN
               X:=CENTRX+DISTANCE;
               Y:=CENTRY-(SIZE DIV 2);
               X1:=CENTRX+RADIUS; Y1:=CENTRY;
               X2:=X1-BONDLEN; Y2:=Y1;
             END;
          3: BEGIN
               X:=CENTRX-(SIZE DIV 2);
               Y:=CENTRY+DISTANCE;
               X1:=CENTRX-2;  Y1:=CENTRY+RADIUS;
               X2:=X1;     Y2:=Y1-BONDLEN;
             END;
          4: BEGIN
               X:=CENTRX-(SIZE DIV 2);
               Y:=CENTRY-DISTANCE-SIZE;
               X1:=CENTRX-2;  Y1:=CENTRY-RADIUS;
```

```
        X2:=X1;     Y2:=Y1+BONDLEN;
       END;
      END; (*CASE*)
     DRAWBLOCK(WATER[J],4,0,0,24,24,X,Y,MODE);
     DRAWLINE(X1,Y1,X2,Y2,WHITE2);
    END;
   END; (*HYDRATE*)

(*-----------------------------------------------------------------*)
PROCEDURE INTROSOLVATE;
(*-----------------------------------------------------------------*)
VAR X,Y:INTEGER; CH:CHAR;
BEGIN
  INITTURTLE;
  X:=10; Y:=YMAX-40;
  WSTAT(X,Y,'This demonstration simplifies the'); Y:=Y-20;
  WSTAT(X,Y,'reaction of ions in solution.'); Y:=Y-50;
  WSTAT(X,Y,'In solution ions are SOLVATED.'); Y:=Y-50;
  WSTAT(X,Y,'In aqueous solution ions are HYDRATED.');
  DEFINEWATER; (*while waiting for <space> set up arrays*)
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
END; (* INTROSOLVATE *)

(*-----------------------------------------------------------------*)
PROCEDURE SHOWHYDRATE;
(*-----------------------------------------------------------------*)
CONST ST='Press <SPACE BAR> to show hydrated ion';
VAR MIDX,MIDY:INTEGER; CH:CHAR;
BEGIN
  INITION(MET,CATION);
  INITTURTLE;
  MIDX:=(XMAX DIV 2)-10; MIDY:=(YMAX DIV 2)+30;
  DRAWBLOCK(CATION,2,0,0,16,16,MIDX-8,MIDY-8,MODE);
  WSTAT(XMIN,YMIN+40,'This represents any metal ion.');
  REPEAT
     DEFINEWATER;
     IF KEYIN THEN READ(CH);
  UNTIL AQUA>WATERNUM;
  WSTAT(XMIN,YMIN,ST);
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
  WSTAT(XMIN,YMIN,ST);
  HYDRATE(WATERNUM,MIDX,MIDY);
  WSTAT(XMIN,YMIN+15,'The number of water molecules involved');
  WSTAT(XMIN,YMIN,'in hydration varies for each ion.');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(SOLVATECATIONS);
END; (* SHOWHYDRATE *)
```

```
(*----------------------------------------------------*)
PROCEDURE NETMETALREACTION;
(*----------------------------------------------------*)
VAR X,Y:INTEGER; CH:CHAR;
BEGIN
 INITTURTLE;
 X:=0; Y:=YMAX-10;
 WSTAT(X,Y,'Net reaction of metal:-'); Y:=Y-40;
 WSTAT(X,Y,'METAL ION'); Y:=Y-10;
 WSTAT(X,Y,'IN ELECTRONIC');
 DRAWARROW(X+100,Y+4,20);
 WSTAT(X+135,Y,'HYDRATED METAL ION'); Y:=Y-10;
 WSTAT(X,Y,' "GLUE" '); Y:=Y-50;
 X:=X+40;
 DRAWBLOCK(ATOM,4,0,0,32,32,X,Y-16,MODE);
 DRAWBLOCK(CATION,2,0,0,16,16,X+8,Y-8,MODE);
 WSTAT(X+12,Y+10,'e');
 X:=X+130;
 DRAWBLOCK(CATION,2,0,0,16,16,X-8,Y-8,MODE);
 HYDRATE(4,X,Y);
 X:=X+65;
 WSTAT(X,Y+4,'e');
 GETKEY(CH,[SPACE,'Q']);
END; (* NETMETALREACTION *)


BEGIN (* SOLVATECATIONS *)
  AQUA:=1;
  INTROSOLVATE;
  SHOWHYDRATE;
  NETMETALREACTION;;
END; (* SOLVATECATIONS *)
```

```
(*AMETAL4*)
  (*==============================================================*)
  PROCEDURE REACTION;
  (*==============================================================*)
  TYPE  ION=RECORD
          X,Y,DX,DY:INTEGER;
        END;
  VAR CYCLE:INTEGER;
    METALX,METALY: INTEGER;
    METALION:ARRAY[1..2] OF ION;

    (*--------------------------------------------------------------*)
    PROCEDURE CHECKKEY;
    (*--------------------------------------------------------------*)
    (*If spacebar (or RETURN) then pause program unti lspacebar is pressed
      again to restart *)
    VAR CH:CHAR;
    BEGIN
      READ(CH);
      IF EOLN(KEYBOARD) THEN CH:=RET;
      IF CH=SPACE THEN
       BEGIN
         CHARTYPE(10);
         WSTAT(186,182,'continue');
         GETKEY(CH,[SPACE,'Q']);
         WSTAT(186,182,'pause    ');
         CHARTYPE(MODE);
       END
        ELSE  QUIT:=((CH='q') OR (CH='Q'));
      IF QUIT THEN EXIT(REACTION);
    END; (* CHECKKEY *)

    (*--------------------------------------------------------------*)
    PROCEDURE DRAWMETAL(SYMBOL:IONTYPE);
    (*--------------------------------------------------------------*)
    VAR  J,LASTX,X,Y : INTEGER;
         SYMB: STRING[2];  ASTR:STRING;  CH:CHAR;
    BEGIN
      X:=16; Y:=1;
      LASTX:=XMAX-32;
      CASE SYMBOL OF
       NA: BEGIN
            SYMB:='Na';
            ASTR:='sodium';
           END;
       CA: BEGIN
            SYMB:='Ca';
            ASTR:='calcium';
           END;
       END; (* CASE *)
      FOR J:=1 TO 2 DO
       BEGIN
        REPEAT
         DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
         MOVETO(X+8,Y+12);
         WSTRING(SYMB);
```

```
     X:=X+32;
    UNTIL (X>=LASTX);
    X:=0; Y:=32;
   END;
  WSTAT(0,184,CONCAT(' The surface atoms of ',ASTR));
  WSTAT(0,174,'are represented by the structure:-');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(REACTION);
  FILLAREA(0,XMAX,172,YMAX,BLACK1);
 END; (* DRAWMETAL *)

(*--------------------------------------------------------------------------*)
PROCEDURE NEWVALUE(ANYNUM: INTEGER);
(*--------------------------------------------------------------------------*)
CONST  INCR=5;
VAR  CENTRX,TOPY,  (*coord. of top centre of targat atom *)
      ANUM:INTEGER;(*determines which atom reacts with acid*)
BEGIN
  CASE ANYNUM OF
  1: BEGIN
       ANUM:=5;
       WITH METALION[1] DO  BEGIN  DX:=-6;  DY:=8;   END;
       WITH METALION[2] DO  BEGIN  DX:=6;  DY:=8;   END;
       WITH WATERMOL[1] DO  BEGIN  DX:=12; DY:=-10; END;
       WITH WATERMOL[2] DO BEGIN   DX:=-4; DY:=-12; END;
    END; (* 1  *)
   2: BEGIN
       ANUM:=3;
       WITH METALION[1] DO  BEGIN  DX:=-6; DY:=8;   END;
       WITH METALION[2] DO  BEGIN  DX:=10; DY:=8;   END;
       WITH WATERMOL[1] DO  BEGIN  DX:=12; DY:=-9; END;
       WITH WATERMOL[2] DO  BEGIN  DX:=-12; DY:=-12; END;
    END; (* 2 *)
   3: BEGIN
       ANUM:=1;
       WITH METALION[1] DO  BEGIN  DX:=-6; DY:=8;   END;
       WITH METALION[2] DO   BEGIN  DX:=10; DY:=8;   END;
       WITH WATERMOL[1] DO  BEGIN  DX:=3;  DY:=-9; END;
       WITH WATERMOL[2] DO  BEGIN   DX:=-12; DY:=-12; END;
    END; (* 3 *)
   4: BEGIN
       ANUM:=3;
       WITH METALION[1] DO  BEGIN  DX:=-6; DY:=10;   END;
       WITH METALION[2] DO  BEGIN  DX:=4; DY:=10;   END;
       WITH WATERMOL[1] DO  BEGIN  DX:=3;  DY:=-9; END;
       WITH WATERMOL[2] DO BEGIN   DX:=-12; DY:=-12; END;
    END; (* 4 *)
  END;(* CASE*)

  METALX:=32*ANUM;
  METALY:=32;
```

```
  CASE METEL OF
   NA: BEGIN
          IF ANYNUM=4 THEN
           BEGIN  METALX:=METALX+16; METALY:=1;   END;
            (*the 4th reaction removes atoms  from lower layer of metal atoms
               other reactions remove atomsfrom top layer of metal atoms*)
         CENTRX:=METALX+32;
        END; (*NA*)
    CA: BEGIN
          IF ANYNUM=4 THEN METALX:=32*(ANUM+1);
          CENTRX:=METALX+16;
       END;
    END; (*CASE*)
  (* centrex is horizontal midpt if one atom is  to be removed or is in between 2
   atoms if 2 atoms removed*)
   (* topy is vertical max. of  metal atom to be   removed ie. y coord + height of
   metal atom *)

  TOPY:=METALY+32;

   (*calculate starting coord of water molecules. One molecule will end up 10
   pixels to the right, the other 10 pixels to the left of centrex. Both  will end up
   10 pixels above atom to be removed*)
  WITH WATERMOL[1] DO
   BEGIN
     X:=CENTRX-42; (* see DEFINESHAPE - proton removed is 22 from
                                                   bottom L.H.C.*)
     Y:=TOPY+10;
     X:=X- (INCR*DX);
     Y:=Y- (INCR*DY);
   END;
  WITH WATERMOL[2] DO
   BEGIN
     X:=CENTRX+10;
     Y:=TOPY+10;
     X:=X- (INCR*DX);
     Y:=Y- (INCR*DY);
   END;
 END; (* NEWVALUE *)

(*----------------------------------------------------------------------*)
PROCEDURE MOVEMOLECULE(VAR ANYREC:MOLECULE; HEIGHT:INTEGER);
(*----------------------------------------------------------------------*)
BEGIN
 WITH ANYREC DO
  BEGIN
    DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
    X:=X+DX;  Y:=Y+DY;
    DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
  END;
END; (*MOVEMOLECULE*)
```

```
(*------------------------------------------------------------------*)
PROCEDURE SHOWWATER(TEMPMODE:INTEGER);
(*------------------------------------------------------------------*)
VAR NUM: INTEGER;
BEGIN
 FOR NUM:=1 TO 2 DO
   BEGIN
     WITH WATERMOL[NUM] DO DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,TEMPMODE);
   END;
END; (* SHOWWATER *)

(*------------------------------------------------------------------*)
PROCEDURE POLARISE;
(*------------------------------------------------------------------*)
(* show +ve charge on H near metal & -ve charge on O *)
(* see DEFINESHAPE to determine x & y values *)
TYPE DASH=PACKED ARRAY[1..2,1..3] OF BOOLEAN;
VAR NUM,MINUSY,PLUSY,MINUSX,PLUSX : INTEGER;
      MINUS: DASH;

      (*..............................................................*)
      PROCEDURE INITMINUS;
      (*..............................................................*)
      VAR S:STRING[3];  COL:INTEGER;
      BEGIN
         S:='XXX';
         FOR COL:=1 TO 3 DO MINUS[1,COL]:=S[COL]='X';
      END; (* INITMUS *)

 BEGIN (*POLARISE*)
  INITMINUS;
  FOR NUM:=1 TO 2 DO
   BEGIN
     MINUSY:=18; PLUSY:=-2;
     CASE NUM OF
        1: BEGIN MINUSX:=22;PLUSX:=18; END;
        2: BEGIN MINUSX:=4; PLUSX:=7; END;
        END; (*CASE*)
     WITH WATERMOL[NUM] DO
      BEGIN
        DRAWBLOCK(MINUS,2,0,0,3,1,X+MINUSX,Y+MINUSY,10);
        WSTAT(X+PLUSX,Y+PLUSY,'+');
      END;
   END; (*FOR*)
  DELAY(250);
 END; (* POLARISE *)

 (*------------------------------------------------------------------*)
 PROCEDURE DISPLAYPROMPT;
 (*------------------------------------------------------------------*)
 VAR CH: CHAR;

      PROCEDURE PROMPT;
      BEGIN
        WSTAT(0,182,'<SPACE BAR> to react metal with water');
      END;
```

```
BEGIN
 PROMPT;
 GETKEY(CH,[SPACE,'Q']);
 IF QUIT THEN EXIT(MICRO);
 PROMPT;
 WSTAT(40,182,'Press <SPACE BAR> to pause');
END; (*DISPLAYPROMPT *)

(*-----------------------------------------------------------------*)
PROCEDURE MOVEWATER;
(*-----------------------------------------------------------------*)
CONST  INCR=5;
 (* required in NEWVALUE to calculate starting position of acid *)
VAR  STEP,NUM: INTEGER;
BEGIN
 FOR STEP:=1 TO INCR DO
   FOR NUM:=1 TO 2 DO
     BEGIN
      MOVEMOLECULE(WATERMOL[NUM],24);
      IF KEYIN THEN
         BEGIN
           CHECKKEY;
           IF QUIT THEN EXIT(MOVEWATER);
         END;
     END;
END; (* MOVEWATER *)

(*-----------------------------------------------------------------*)
PROCEDURE MOVEHYDROX(NUM:INTEGER);
(*-----------------------------------------------------------------*)
BEGIN
 MOVEMOLECULE(HYDROXIDE[NUM],16);
END; (* MOVEWATER *)

(*-----------------------------------------------------------------*)
PROCEDURE REACT;
(*-----------------------------------------------------------------*)
VAR   I,PROTONX,PROTONY,NUMIONS : INTEGER;
      (*..........................................................*)
      PROCEDURE CALCULATE;
      (*..........................................................*)
      (*Calculate starting coordinates of two hydroxide ions according to final
      position  of reacting water molecules.  Direction of hydroxides is opposite to
      the direction of the water molecules.Also calculate the coordinates
      of the proton in the left water molecule which reacts.  The other proton
      has the same y coord. but is 20 units to right- see NEWVALUE  *)
      VAR NUM:INTEGER;
      BEGIN
       FOR NUM:=1 TO 2 DO
         WITH HYDROXIDE[NUM] DO
          BEGIN
             X:=WATERMOL[NUM].X;
             Y:=WATERMOL[NUM].Y+10; (*see DEFINESHAPE *)
             DX:=-WATERMOL[NUM].DX;
             DY:=-WATERMOL[NUM].DY;
```

```
    END;
 PROTONX:=WATERMOL[1].X+22; (*see DEFINESHAPE*)
                              (* coord of left proton*)
 PROTONY:=WATERMOL[1].Y-6;
END;(* CALCULATE *) (*spacing of 20 between protons*)

(*.............................................................*)
PROCEDURE IONIZE(XX,YY:INTEGER; VAR ANUM:INTEGER);
(*.............................................................*)
(* EXCHANGE  ATOM(S) FOR  IONS  & INITIALISE ION RECORD*)
VAR NUM:INTEGER;
BEGIN
 FOR NUM:=1 TO ANUM DO
  BEGIN
    IF NUM=2 THEN XX:=XX+32;    (*erase atom*)
    DRAWBLOCK(ATOM,4,0,0,32,32,XX,YY,4);
    WITH METALION[NUM] DO
     BEGIN
       X:=XX +8; Y:=YY +8;
       DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
     END;               (* display ion *)
  END; (*FOR*)
END; (* IONIZE  *)

(*.............................................................*)
PROCEDURE DISPLAYHYDROX;
(*.............................................................*)
VAR NUM  : INTEGER;
BEGIN
 FOR NUM:=1 TO 2 DO
   BEGIN
    WITH HYDROXIDE[NUM] DO DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,MODE);
   END;
END; (* DISPLAYHYDROX *)

(*.............................................................*)
PROCEDURE SHOWPROTONS(X,Y:INTEGER);
(*.............................................................*)
BEGIN
 WSTAT(X-8,Y,'+');
 DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
 DRAWBLOCK(HATOM,2,0,0,5,5,X+25,Y,MODE);
 WSTAT(X+31,Y,'+');
END; (* SHOWPROTONS *)

(*.............................................................*)
PROCEDURE SHOWELECTRONS(X,Y:INTEGER);
(*.............................................................*)
VAR J: INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
  BEGIN
    WSTAT(X+8,Y-1,'e');
    X:=X+8;
  END;
END;  (* SHOWELECTRONS *)
```

```
(*..........................................................................*)
PROCEDURE SHOWHYDROGEN(X,Y:INTEGER);
(*..........................................................................*)
BEGIN
 DRAWBLOCK(HYDROGEN,2,0,0,16,5,X,Y,MODE);
END;

(*..........................................................................*)
PROCEDURE MOVEH2(VAR X,Y:INTEGER);
(*..........................................................................*)
BEGIN
 SHOWHYDROGEN(X,Y);  (* erase H2 *)
 Y:=Y+8;
 SHOWHYDROGEN(X,Y);  (* display H2 *)
END; (* MOVEH2 *)

(*..........................................................................*)
PROCEDURE MOVENAIONS(VAR NUM:INTEGER);
(*..........................................................................*)
VAR   ANUM : INTEGER;
BEGIN
 FOR ANUM:=1 TO NUM DO
   BEGIN
    WITH METALION[ANUM]  DO
      BEGIN                   (*erase*)
         DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
         X:=X+DX; Y:=Y+DY;
         DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
      END;
   END;(*  FOR  *)
END;  (* MOVENAIONS *)

(*..........................................................................*)
PROCEDURE ERASENAIONS(NUM:INTEGER);
(*..........................................................................*)
VAR ANUM:INTEGER;
BEGIN
 FOR ANUM:=1  TO NUM DO
 WITH METALION[ANUM] DO
     DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);(*erase*)
END;  (* ERASENAIONS *)


BEGIN  (* REACT *)
   CASE METEL OF
    NA: NUMIONS:=2;
    CA: NUMIONS:=1;
    END; (*CASE*)
   POLARISE;
   CALCULATE;  (* starting coord of OH- from final coord of H2O *)
   IF KEYIN THEN CHECKKEY;
   SHOWWATER(0);  (*erase both H2O from current position*)
   DISPLAYHYDROX;  (*display 2 OH- in previous position of H2O*)
   IONIZE(METALX,METALY,NUMIONS);(* replace atom with ion*)
   SHOWPROTONS(PROTONX,PROTONY); (* display 2 protons*)
   SHOWELECTRONS(PROTONX,PROTONY);(* display 2 electrons*)
```

```
        DELAY(300);
        IF KEYIN THEN CHECKKEY;
        SHOWPROTONS(PROTONX,PROTONY);  (* erase 2 protons *)
        SHOWELECTRONS(PROTONX,PROTONY); (* erase 2 electrons*)
        PROTONX:=PROTONX+7; (* H2 is 16 bits wide whereas  H+-H+ was
                        30 bits wide therefore move across 7 bits to centre *)
        SHOWHYDROGEN(PROTONX,PROTONY);(* display H2  *)
        IF KEYIN THEN CHECKKEY;
        FOR I:=1 TO 4 DO
          BEGIN       (* move OH- & H2 *)
             MOVEHYDROX(1);
             MOVEHYDROX(2);
             MOVEH2(PROTONX,PROTONY);
             IF KEYIN THEN CHECKKEY;
          END;
        FOR I:=1 TO 12 DO
          BEGIN        (* move OH-, H2  & cations *)
             IF ODD(I) THEN MOVEHYDROX(1) ELSE MOVEHYDROX(2);
             MOVEH2(PROTONX,PROTONY);
             MOVENAIONS(NUMIONS);
             IF KEYIN THEN CHECKKEY;
          END;
        DISPLAYHYDROX;   (* erase OH- *)
        SHOWHYDROGEN(PROTONX,PROTONY);  (*erase H2 *)
        ERASENAIONS(NUMIONS);      (* erase cations *)
      END; (*REACT*)

    BEGIN  (*REACTION*)
     FOR METEL:=NA TO CA DO
       IF NOT QUIT THEN
         BEGIN
           INITION(METEL,CATION);
           INITTURTLE;
           DRAWMETAL(METEL);
           CYCLE:=0;
           REPEAT
             CYCLE:=CYCLE+1;
             NEWVALUE(CYCLE);
             SHOWWATER(MODE);
             IF CYCLE=1 THEN DISPLAYPROMPT; (*start reaction*)
             MOVEWATER;
             IF NOT QUIT THEN REACT;
           UNTIL ((CYCLE=4) OR QUIT);
                    (* H20  reacts 4 times *)
         END;
    END; (*REACTION*)
```

```
(* AMETALS *)
  (*============================================================*)
  PROCEDURE CONCLUSION;
  (*============================================================*)
  CONST  X1=1;Y1=80;

    (*------------------------------------------------------------*)
    PROCEDURE DRAWPLUS(X,Y:INTEGER);
    (*------------------------------------------------------------*)
    CONST  SIZE=10;
    BEGIN
      MOVECOL(X,Y,WHITE1);
      MOVECOL(X+SIZE,Y,NONE);
      X:=X+(SIZE DIV 2);  Y:=Y+(SIZE DIV 2);
      MOVECOL(X,Y,WHITE1);
      MOVECOL(X,Y-SIZE,NONE);
    END;(*DRAWPLUS*)

    (*------------------------------------------------------------*)
    PROCEDURE DRAWH2(X,Y: INTEGER);
    (*------------------------------------------------------------*)
    BEGIN
      (* DRAWBOND *)
      DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
      DRAWBLOCK(HATOM,2,0,0,5,5,X,Y+15,MODE);
      MOVECOL(X+3,Y+7,WHITE1);MOVECOL(X+3,Y+12,NONE);
    END; (* DRAWH2 *)

    (*------------------------------------------------------------*)
    PROCEDURE NETMETAL;
    (*------------------------------------------------------------*)
    VAR X,Y:INTEGER;
       SYMBOL:STRING[2];
    BEGIN
      Y:=YMAX-10;X:=1;
      INITTURTLE;
      WSTAT(X,Y,'Net reaction of metal:-');
      FOR METEL:=NA TO CA DO
       BEGIN
        INITION(METEL,CATION);
        X:=1;
        CASE METEL OF
          NA:BEGIN SYMBOL:='Na'; Y:=YMAX-60; END;
          CA:BEGIN SYMBOL:='Ca'; Y:=YMAX-120;END;
          END;(*CASE*)
        DRAWBLOCK(ATOM,4,0,0,32,32,X,Y,MODE);
        WSTAT(X+8,Y+12,SYMBOL);
        X:=X+70;
        DRAWARROW(X,Y+16,30);
        X:=X+80;
        DRAWBLOCK(CATION,2,0,0,16,16,X,Y+8,MODE);
        X:=X+50;
        DRAWPLUS(X,Y+16);
        X:=X+30;
        WSTAT(X,Y+12,'e');
        IF METEL=CA THEN WSTAT(X+20,Y+12,'e');
```

```
    Y:=Y-60;
   END; (* FOR *)
END; (* NETMETAL *)

(*------------------------------------------------------------------------*)
PROCEDURE NETWATER;
(*------------------------------------------------------------------------*)
VAR X,Y:INTEGER;
BEGIN
 X:=1; Y:=YMAX-10;
 INITTURTLE;
 WSTAT(X,Y,'Net reaction of water:-');
 Y:=Y-80;
 DRAWBLOCK(WATERMOL[2].SHAPE,4,0,0,32,24,X,Y-32,MODE);
 DRAWBLOCK(WATERMOL[2].SHAPE,4,0,0,32,24,X,Y+12,MODE);
 X:=X+50;
 DRAWPLUS(X,Y);
 X:=X+30;
 WSTAT(X,Y,'e');
 WSTAT(X+20,Y,'e');
 X:=X+50;
 DRAWARROW(X,Y,30);
 X:=X+50;
 DRAWH2(X,Y-8);
 X:=X+20;
 DRAWPLUS(X,Y);
 X:=X+30;
 DRAWBLOCK(HYDROXIDE[2].SHAPE,4,0,0,32,16,X,Y-35,MODE);
 DRAWBLOCK(HYDROXIDE[2].SHAPE,4,0,0,32,16,X,Y+15,MODE);
END; (* NETHYDROGEN *)

(*------------------------------------------------------------------------*)
PROCEDURE NETREACTION(METEL: IONTYPE);
(*------------------------------------------------------------------------*)
VAR SYMBOL:STRING[2]; AMETL:STRING[10];
   K,X,Y:INTEGER;
   UNITCHARGE:BOOLEAN;

     PROCEDURE DRAWMETAL(METX,METY:INTEGER);
     BEGIN
      DRAWBLOCK(ATOM,4,0,0,32,32,METX,METY,MODE);
      WSTAT(METX+8,METY+12,SYMBOL);
     END;

BEGIN (* NETREACTION *)
 UNITCHARGE:=METEL=NA;
 CASE METEL OF
 NA: BEGIN
       INITION(NA,CATION);
       SYMBOL:='Na'; AMETL:='sodium:-';
     END;
 CA: BEGIN
       INITION(CA,CATION);
       SYMBOL:='Ca'; AMETL:='calcium:-';
      END;
 END; (*CASE*)
```

```
INITTURTLE;
X:=X1; Y:=YMAX-10;
WSTAT(X,Y,CONCAT('Reaction between water and ',AMETL));
Y:=Y-100;
IF UNITCHARGE THEN
 BEGIN
   DRAWMETAL(X,Y);
   IF UNITCHARGE THEN DRAWMETAL(X,Y+40);
 END
    ELSE DRAWMETAL(X,Y+20);
Y:=Y1; X:=X1+60;

DRAWBLOCK(WATERMOL[2].SHAPE,4,0,0,32,24,X,Y+8,MODE);
DRAWBLOCK(WATERMOL[2].SHAPE,4,0,0,32,24,X,Y+48,MODE);
X:=X1+155;
IF UNITCHARGE THEN
 BEGIN
   DRAWBLOCK(CATION,2,0,0,16,16,X,Y+8,MODE);
   DRAWBLOCK(CATION,2,0,0,16,16,X,Y+56,MODE);
 END
     ELSE DRAWBLOCK(CATION,2,0,0,16,16,X,Y+28,MODE);

X:=X1+245;
DRAWBLOCK(HYDROXIDE[2].SHAPE,4,0,0,32,16,X,Y+8,MODE);
DRAWBLOCK(HYDROXIDE[2].SHAPE,4,0,0,32,16,X,Y+56,MODE);

X:=X1+216;
DRAWH2(X,Y+30);

DRAWARROW(X1+105,Y+36,30);
DRAWPLUS(X1+40,Y+36); DRAWPLUS(X1+185,Y+36);
DRAWPLUS(X1+240,Y+36);
END; (* NETREACTION *)

(*-----------------------------------------------------------------*)
PROCEDURE CHECKKEY;
(*-----------------------------------------------------------------*)
VAR CH:CHAR;
BEGIN
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(CONCLUSION);
END; (* CHECKKEY *)

BEGIN
  NETMETAL;
  CHECKKEY;
  NETWATER;
  CHECKKEY;
  NETREACTION(NA);
  CHECKKEY;
  NETREACTION(CA);
END; (* CONCLUSION *)
```

```
BEGIN  (* MICRO *)
  INITSHAPES;
  EXPLAINSHAPES;
  REPEAT
     IF NOT QUIT THEN REACTION;
     IF NOT QUIT THEN CONCLUSION;
     IF NOT QUIT THEN SOLVATECATIONS;
     PAGE(OUTPUT);
  UNTIL FIN('MICRO');
END; (* MICRO *)


(* AMETAL6 *)
(*************************************************************)
PROCEDURE MACRO;
(*************************************************************)
CONST  BEAKERY=36;   BSIZE=96;
TYPE
   BITSHAPE=PACKED ARRAY[1..6,1..16]OF BOOLEAN;
   METTYPE=(NA,CA);
VAR
   MET:METTYPE;
   METSHAPE:INTEGER;                (*index to shape of metal *)
   METAL:ARRAY[1..3] OF BITSHAPE;        (* shapes of metal *)
   LEVEL,                             (* ht of soln in beaker *)
   BEAKERX,                           (* position of beaker   *)
   METDX,
   METALX, METALY:INTEGER;             (* position of metal    *)
   SPACEPR:BOOLEAN;

    (*===========================================================*)
    PROCEDURE DRAWBEAKER(X,Y,SIZE:INTEGER; COL:SCREENCOLOR);
    (*===========================================================*)
    (* x,y is coord for the bottom L.H.corner of beaker *)
    VAR  EDGE  : INTEGER;
    BEGIN
      EDGE:=SIZE DIV 12;
      MOVECOL(X-EDGE,Y+SIZE+EDGE,COL);
      MOVETO(X,Y+SIZE);
      MOVETO(X,Y);
      MOVETO(X+SIZE,Y);
      MOVETO(X+SIZE,Y+SIZE);
      MOVECOL(X+SIZE+EDGE,Y+SIZE+EDGE,NONE);
    END; (* DRAWBEAKER *)

    (*===========================================================*)
    PROCEDURE FILLBEAKER(X,Y,SIZE:INTEGER);
    (*===========================================================*)
    BEGIN
     DRAWLINE(X,Y,X+SIZE,Y,WHITE1);
    END;
```

```
(*==============================================================*)
PROCEDURE INITBITS(ROW:INTEGER;VAR BITS: BITSHAPE;S:STRING);
(*==============================================================*)
VAR COL:INTEGER;
BEGIN
 FOR COL:=1 TO 16 DO  BITS[ROW,COL]:=S[COL]='X';
END; (* INITBIT *)

(*==============================================================*)
PROCEDURE INITMETAL;
(*==============================================================*)
(* initialize shapes of metal *)
BEGIN
  INITBITS(6,METAL[1],' XXX   XXXXXX   ');
  INITBITS(5,METAL[1],' XXXX XXXXXXXXXX');
  INITBITS(4,METAL[1],'XXXXXXXXXXXXXXXX');
  INITBITS(3,METAL[1],'XXXXXXXXXXXXXXXX');
  INITBITS(2,METAL[1],'XXXXXXXXXXXXXXXX');
  INITBITS(1,METAL[1],' XXXXXXXX  XXXX ');


  INITBITS(6,METAL[2],'                ');
  INITBITS(5,METAL[2],'                ');
  INITBITS(4,METAL[2],'    XX    XXX   ');
  INITBITS(3,METAL[2],'XXXXXXXXXXXXXX  ');
  INITBITS(2,METAL[2],' XXXXXXXXXXXX   ');
  INITBITS(1,METAL[2],'   XXXXXXXXXX   ');

  INITBITS(6,METAL[3],'                ');
  INITBITS(5,METAL[3],'                ');
  INITBITS(4,METAL[3],' XX XX          ');
  INITBITS(3,METAL[3],'XXXXXXXX        ');
  INITBITS(2,METAL[3],' XXXXXXXX       ');
  INITBITS(1,METAL[3],' XXXXXXXX       ');
 END; (* INITMETAL *)

 (*=============================================================*)
 PROCEDURE INITVARS;
 (*=============================================================*)
 BEGIN
   BEAKERX:=140-(BSIZE DIV 2);
   LEVEL:=BEAKERY + (2*BSIZE DIV 3);
    INITMETAL;
 END; (* INITVARS *)

 (*=============================================================*)
 PROCEDURE DROPMETAL(VAR X,Y:INTEGER; AMET:METTYPE);
 (*=============================================================*)
 VAR CH:CHAR;  BOTTOM:INTEGER;
    S:STRING[10];
```

```
    (*------------------------------------------------------------------*)
    PROCEDURE REQUEST;
    (*------------------------------------------------------------------*)
    BEGIN
      WSTAT(160,Y,S);
      WSTAT(30,5,'Press <SPACE BAR> to add metal');
    END; (*REQUEST*)

  BEGIN (*DROPMETAL*)
   CASE AMET OF
     NA: BEGIN S:='Sodium'; BOTTOM:=LEVEL; END;
     CA: BEGIN S:='Calcium'; BOTTOM:=BEAKERY+8; END;
    END; (*CASE *)
   DRAWBLOCK(METAL[1],2,0,0,16,6,X,Y,MODE); (*display metal*)
   REQUEST;
   GETKEY(CH,[SPACE,'Q']);
   IF QUIT THEN EXIT(MACRO);
   REQUEST;
   REPEAT
     DRAWBLOCK(METAL[1],2,0,0,16,6,X,Y,MODE);    (* erase *)
     Y:=Y-10;    (* CALC. NEW HEIGHT*)
     IF ((Y<=LEVEL) AND (AMET=NA)) THEN
       DRAWLINE(X,LEVEL,X+16,LEVEL,BLACK1);
     DRAWBLOCK(METAL[1],2,0,0,16,6,X,Y,MODE);    (* display *)
     DELAY(30);
   UNTIL Y<=(BOTTOM);
  END; (* DROPMETAL *)

  (*=================================================================*)
  PROCEDURE REACTION;
  (*=================================================================*)

     (*--------------------------------------------------------------*)
     PROCEDURE STATEMENT;
     (*--------------------------------------------------------------*)
     VAR S1,S2:STRING;
     BEGIN
       S1:='Sodium dissolves in water';
       S2:='and a gas is evolved.';
       WSTAT(30,13,S1);
       WSTAT(40,1,S2);
     END; (* STATEMENT *)

     (*--------------------------------------------------------------*)
     PROCEDURE GAS;
     (*--------------------------------------------------------------*)
     VAR S1,S2:STRING;
     BEGIN
       S1:='Gas produced was explosive- ';
       S2:='this indicates hydrogen gas.';
       WSTAT(30,13,S1);
       WSTAT(30,1,S2);
     END; (* GAS *)
```

```
(*------------------------------------------------------------------*)
PROCEDURE LITMUSTEST;
(*------------------------------------------------------------------*)
VAR CH:CHAR;
  (*................................................................*)
  PROCEDURE TESTSOLN(X,SIZE,ALEVEL:INTEGER);
  (*................................................................*)
  TYPE  SMSIZE=PACKED ARRAY[1..8,1..8] OF BOOLEAN;
  VAR DROP : SMSIZE;  CH:CHAR;
      Y:INTEGER;  (* y coord of litmus drop *)

    PROCEDURE INITDROP;
        PROCEDURE SMALLBITS(ROW : INTEGER;  VAR BITS:SMSIZE;S:STRING);
        VAR  COL: INTEGER;
        BEGIN
         FOR COL:=1 TO 8 DO  BITS[ROW,COL]:=S[COL]='X';
        END;  (* SMALLBITS *)

     BEGIN  (* INITDROP *)
       SMALLBITS(8,DROP,'   X    ');
       SMALLBITS(7,DROP,'  XXX   ');
       SMALLBITS(6,DROP,' XXXXX  ');
       SMALLBITS(5,DROP,'XXXXXXX ');
       SMALLBITS(4,DROP,'XXXXXXXX');
       SMALLBITS(3,DROP,'XXXXXXXX');
       SMALLBITS(2,DROP,' XXXXXX ');
       SMALLBITS(1,DROP,'  XXXX  ');
       (*init starting coordinates of drop*)
       X:=X+(BSIZE DIV 2);   (*centre of beaker*)
       Y:=YMAX-10;           (*near top of screen*)
     END; (* INITDROP *)

    PROCEDURE MOVEDROP(VAR X,Y:INTEGER);
    BEGIN
     REPEAT
       DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);(*erase *)
       Y:=Y-6;
       DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);(*display *)
       DELAY(30);
     UNTIL (Y<LEVEL);
     DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);(*erase *)
    END;  (* MOVEDROP *)

  BEGIN  (* TESTSOLN *)
    INITDROP;
    DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);
    MOVEDROP(X,Y);
    FILLAREA(BEAKERX+2,BEAKERX+BSIZE,BEAKERY+1,LEVEL,BLUE);
  END; (* TESTSOLN *)

  (*................................................................*)
  PROCEDURE REQUEST;
  (*................................................................*)
  BEGIN
   WSTAT(30,183,'Press <SPACE BAR> to add litmus');
  END;
```

```
  (*...........................................................................*)
  PROCEDURE RESULT;
  (*...........................................................................*)
  BEGIN
    WSTAT(60,13,'Litmus turned blue.');
    WSTAT(60,1,'Solution is basic.');
  END;

BEGIN (* LITMUSTEST *)
  REQUEST;
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(LITMUSTEST);
  REQUEST; (*erase*)
  GAS; (*erase*)
  TESTSOLN(BEAKERX,BSIZE,LEVEL);
  RESULT;
  GETKEY(CH,[SPACE,'Q']);
  RESULT;
END; (* LITMUSTEST *)

(*---------------------------------------------------------------------------*)
PROCEDURE MOVEMETAL(VAR ANYARRAY:BITSHAPE; VAR X,Y,DX:INTEGER);
(*---------------------------------------------------------------------------*)
VAR WIDTH: INTEGER;

  (*...........................................................................*)
  PROCEDURE CHANGEDIRECTION(VAR Z,DZ: INTEGER);
  (*...........................................................................*)
  BEGIN
    Z:=Z-DZ;
    DZ:=-DZ;
  END; (* CHANGEDIRECTION *)

 BEGIN
  IF ANYARRAY=METAL[3] THEN WIDTH:=9 ELSE WIDTH:=15;
  DRAWBLOCK(ANYARRAY,2,0,0,16,6,X,Y,MODE);  (* erase *)
  DRAWLINE(X,LEVEL,X+WIDTH,LEVEL,WHITE1);
  X:=X+DX;
  IF (X<=BEAKERX) OR (X>(BEAKERX+BSIZE-16)) THEN
       CHANGEDIRECTION(X,DX);
  DRAWLINE(X,LEVEL,X+WIDTH,LEVEL,BLACK1);
  DRAWBLOCK(ANYARRAY,2,0,0,16,6,X,Y,MODE);
                  (* display at new position*)
 END; (* MOVEMETAL *)

(*---------------------------------------------------------------------------*)
PROCEDURE EXPLOSION(CX,CY:INTEGER; COL:SCREENCOLOR);
(*---------------------------------------------------------------------------*)
CONST DIST=12;

  (*...........................................................................*)
  PROCEDURE SHOWSLASH(ANGLE:INTEGER);
  (*...........................................................................*)
  BEGIN
    MOVETO(CX,CY);
    TURNTO(ANGLE);
```

```
      MOVE(DIST);
      PENCOLOR(COL);
      MOVE(DIST);
      PENCOLOR(NONE);
    END;(*SHOWSLASH*)

  BEGIN (* EXPLOSION *)
    SHOWSLASH(0);
    SHOWSLASH(45);
    SHOWSLASH(90);
    SHOWSLASH(135);
    SHOWSLASH(180);
    SHOWSLASH(225);
    SHOWSLASH(315);
  END; (* EXPLOSION *)

(*-----------------------------------------------------------------*)
PROCEDURE CHECKKEY(VAR SP:BOOLEAN);
(*-----------------------------------------------------------------*)
VAR CH:CHAR;
BEGIN
  READ(CH);
  QUIT :=((CH='Q') OR (CH='q'));
  SP:=CH=SPACE;
END; (* CHECKKEY *)

(*-----------------------------------------------------------------*)
PROCEDURE CYCLE(SHAPE,NUMCYCLES:INTEGER);
(*-----------------------------------------------------------------*)
CONST SPEED=30;
VAR CYCLES:INTEGER;
BEGIN
  CYCLES:=0;
  REPEAT
    MOVEMETAL(METAL[SHAPE],METALX,METALY,METDX);
    DELAY(SPEED);
    IF KEYIN THEN CHECKKEY(SPACEPR);
    CYCLES:=CYCLES+1;
  UNTIL ((CYCLES>=NUMCYCLES) OR QUIT OR SPACEPR);
  IF QUIT THEN EXIT(REACTION);
END; (* CYCLE *)

(*-----------------------------------------------------------------*)
PROCEDURE SHOWEXPLOSION(CURRENTX:INTEGER);
(*-----------------------------------------------------------------*)
BEGIN
  EXPLOSION(CURRENTX,LEVEL+20,WHITE2);
  IF MET=NA THEN CYCLE(METSHAPE,2);
  EXPLOSION(CURRENTX,LEVEL+20,BLACK2);
  IF KEYIN THEN CHECKKEY(SPACEPR);
  IF QUIT THEN EXIT(REACTION);
END; (* SHOWEXPLOSION *)
```

```
(*-----------------------------------------------------------------*)
PROCEDURE SWAPMETAL(VAR CURRENT:INTEGER);
(*-----------------------------------------------------------------*)
BEGIN
  DRAWBLOCK(METAL[CURRENT],2,0,0,16,6,METALX,METALY,MODE);
  CURRENT:=CURRENT+1;
  DRAWBLOCK(METAL[CURRENT],2,0,0,16,6,METALX,METALY,MODE);
END; (*SWAPMETAL*)

(*-----------------------------------------------------------------*)
PROCEDURE KEEPDISSOLVING;
(*-----------------------------------------------------------------*)
VAR J:INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
  BEGIN
   CYCLE(METSHAPE,20);
   IF SPACEPR THEN EXIT(KEEPDISSOLVING);
   SWAPMETAL(METSHAPE);
  END;
 DRAWLINE(METALX+10,LEVEL,METALX+15,LEVEL,WHITE1);

 FOR J:=1 TO 3 DO
  BEGIN
   CYCLE(METSHAPE,10);
   IF SPACEPR THEN EXIT(KEEPDISSOLVING);
  END;
END; (* KEEPDISSOLVING *)

(*-----------------------------------------------------------------*)
PROCEDURE SODIUMREACTION;
(*-----------------------------------------------------------------*)
BEGIN
 STATEMENT;
 METSHAPE:=1;
 METDX:=8;
 CYCLE(METSHAPE,10);

 SPACEPR:=FALSE;
 SHOWEXPLOSION(METALX);

 KEEPDISSOLVING;
 DRAWBLOCK(METAL[METSHAPE],2,0,0,16,6,METALX,METALY,MODE);
                    (*erase metal*)
 DRAWLINE(METALX,LEVEL,METALX+15,LEVEL,WHITE1); (*fix up surface *)

 STATEMENT;
END; (* SODIUMREACTION *)
```

```
(* AMETAL7 *)
    (*-----------------------------------------------------------------*)
    PROCEDURE CALCIUMREACTION;
    (*-----------------------------------------------------------------*)
    TYPE
      BUBSHAPE=PACKED ARRAY[0..5,0..7]OF BOOLEAN;
      FLSHAPE=PACKED ARRAY[0..7,0..7]OF BOOLEAN;
    VAR
      BUBBLE:BUBSHAPE;          (* shape of bubbles    *)
      FLAME:FLSHAPE;            (* shape of flame      *)
      LOWLEVEL,MATCHX,MATCHY,
      BUBLX,BUBLY,              (*position of top bubble *)
      SPEED,
      DY,     (* required for movebubbles - determines no.
              pixels that each bubble rises in one cycle *)
      GAP,                      (*spacing between bubbles*)
      TOPLEVEL,J :INTEGER;

      (*..............................................................*)
      PROCEDURE INIT;
      (*..............................................................*)
      CONST SKIP = 4; (*determines GAP between bubbles*)

        PROCEDURE INITBUBBLE;
              PROCEDURE INIT(ROW:INTEGER;VAR BITS:BUBSHAPE;S:STRING);
              VAR  COL:INTEGER;
              BEGIN
                 FOR COL:=0 TO 7 DO  BITS[ROW,COL]:=S[COL+1]='X';
              END; (* INITBIT *)

         BEGIN (* INITBUBBLE *)
           INIT(5,BUBBLE,'  XX   ');
           INIT(4,BUBBLE,' X X  ');
           INIT(3,BUBBLE,'X    X ');
           INIT(2,BUBBLE,'X    X ');
           INIT(1,BUBBLE,' X X  ');
           INIT(0,BUBBLE,'  XX   ');
         END; (* INITBUBBLE *)

        PROCEDURE INITFLAME;
            PROCEDURE INITFL(ROW:INTEGER;VAR BITS:FLSHAPE; S:STRING);
            VAR  COL:INTEGER;
            BEGIN
               FOR COL:=0 TO 7 DO   BITS[ROW,COL]:=S[COL+1]='X';
            END; (* INITBIT *)

         BEGIN (* INITFLAME  *)
           INITFL(7,FLAME,'  XX   ');
           INITFL(6,FLAME,'  XX   ');
           INITFL(5,FLAME,' X X  ');
           INITFL(4,FLAME,' X X  ');
           INITFL(3,FLAME,'X    X ');
           INITFL(2,FLAME,'X    X ');
           INITFL(1,FLAME,' X X  ');
           INITFL(0,FLAME,'  XX   ');
         END; (* INITFLAME  *)
```

```
BEGIN (* INIT *)
  TOPLEVEL:=LEVEL-2;
  LOWLEVEL:=BEAKERY+8;
  BUBLX:=BEAKERX+(BSIZE DIV 2);
  BUBLY:=BEAKERY+6;
  MATCHX:=62;MATCHY:=160;
  DY:=4;
  SPEED:=35;(*required for movebubbles - determines time between each cycle
               in which every bubble is moved up by DY pixels*)
  GAP:=SKIP*DY; (*spacing between each bubble in the row - this is an integral
                                                                 no. of DY*)
  INITBUBBLE;
  INITFLAME;
END; (* INIT *)

(*.......................................................................*)
PROCEDURE HEATTUBE;
(*.......................................................................*)
VAR INITX,INITY,DX :INTEGER;CH:CHAR;
    S:STRING;
    BURNER: PACKED ARRAY[0..7,0..15] OF BOOLEAN;

    PROCEDURE INITBUNSEN;
        PROCEDURE INITBURNER(ROW:INTEGER;S:STRING);
        VAR COL:INTEGER;
        BEGIN
           FOR COL:=0 TO 15 DO BURNER[ROW,COL]:=S[COL+1]='X';
        END; (* INITBURNER *)

    BEGIN (*INITBUNSEN*)
       INITBURNER(7,'       XX       ');
       INITBURNER(6,'      XX X      ');
       INITBURNER(5,'    XX    XX    ');
       INITBURNER(4,'   XX      XX   ');
       INITBURNER(3,'  XX        XX  ');
       INITBURNER(2,'  XX        XX  ');
       INITBURNER(1,'   XX      XX   ');
       INITBURNER(0,'     XXXXXX     ');
    END; (* INITBENSEN *)

 BEGIN (* HEATTUBE *)
    INITBUNSEN;
    S:='Press <SPACE BAR> to heat.';
    WSTAT(30,5,S);
    GETKEY(CH,[SPACE,'Q']);
    IF QUIT THEN EXIT(HEATTUBE);
    WSTAT(30,5,S);
    DRAWBLOCK(BURNER,2,0,0,16,8,METALX,BEAKERY-16,MODE);
    DELAY(1500);                    (*display*)
    DRAWBLOCK(BURNER,2,0,0,16,8,METALX,BEAKERY-16,MODE);
 END; (* HEATTUBE *)                  (*erase*)
```

```
(*.....................................................................*)
PROCEDURE DRAWMATCH(X,Y:INTEGER; COL:SCREENCOLOR);
(*.....................................................................*)
BEGIN
 MOVECOL(X-12,Y+8,COL);
 MOVETO(X,Y);
 MOVECOL(X,Y+4,NONE);
 DRAWBLOCK(FLAME,2,0,0,8,8,X-3,Y+6,MODE);
END; (* DRAWMATCH *)

(*.....................................................................*)
PROCEDURE MOVEMATCH(VAR MATCHX,MATCHY:INTEGER);
(*.....................................................................*)
BEGIN
 DRAWMATCH(MATCHX,MATCHY,BLACK2);
 MATCHX:=MATCHX+8;
 MATCHY:=MATCHY-3;
 DRAWMATCH(MATCHX,MATCHY,WHITE2);
END; (* MOVEMATCH *)

(*.....................................................................*)
PROCEDURE REQUEST;
(*.....................................................................*)
BEGIN
 WSTAT(30,183,'Press <SPACE BAR> to test gas');
END; (* REQUEST *)

(*.....................................................................*)
PROCEDURE DRAWBUBBLES(X,Y:INTEGER);
(*.....................................................................*)
(*draws one bubble , then draws a row of extra bubbles underneath*)

    PROCEDURE EXTRABUBBLES(X,NEWY:INTEGER);
    (*draws as many bubbles as possible with a spacing of GAP below the top
     bubble until LOWLEVEL is reached*)
    VAR MORE:BOOLEAN;
    BEGIN
     REPEAT
       NEWY:=NEWY-GAP;
       MORE:=NEWY>=LOWLEVEL;
       IF MORE  THEN  DRAWBLOCK(BUBBLE,2,0,0,8,6,X,NEWY,MODE)
     UNTIL NOT MORE;
    END; (* EXTRABUBBLES *)

BEGIN (* DRAWBUBBLES *)
 DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE);
 EXTRABUBBLES(X,Y);
END; (* DRAWBUBBLES *)

(*.....................................................................*)
PROCEDURE MOVEBUBBLES(VAR X,CURRENTY:INTEGER);
(*.....................................................................*)
VAR Y:INTEGER;
BEGIN  (* MOVEBUBBLES *)
 Y:=CURRENTY;
 (*starting at top of row erase each bubble at current position and redraw
```

```
  at DY pixels higher up - unless the bubble rises above LEVEL of soln in which
  it is not redrawn. Each bubble is separated by GAP pixels*)
 REPEAT
   DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE);     (*erase *)
   IF (Y+DY)<TOPLEVEL THEN
   DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE); (*display *)
   Y:=Y-GAP;
 UNTIL Y<LOWLEVEL;

  IF (Y+DY)>=LOWLEVEL THEN DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE);
                  (*display new bubble at bottom*)
  CURRENTY:=CURRENTY+DY;
  IF CURRENTY>=TOPLEVEL THEN CURRENTY:=CURRENTY-GAP;
END; (* MOVEBUBBLES  *)

(*.......................................................................................*)
PROCEDURE BUBCYCLE(CYCLENUM:INTEGER);
(*.......................................................................................*)
VAR CYCLES:INTEGER;
BEGIN
 CYCLES:=0;
  REPEAT
     CYCLES:=CYCLES+1;
     MOVEBUBBLES(BUBLX,BUBLY);
     DELAY(SPEED);
     IF KEYIN THEN CHECKKEY(SPACEPR);
  UNTIL ((CYCLES>=CYCLENUM) OR QUIT OR SPACEPR);
  IF QUIT THEN EXIT(REACTION);
 END; (* BUBCYCLE *)


BEGIN (* CALCIUMREACTION *)
 INIT;
 IF NOT QUIT THEN HEATTUBE;
 SPACEPR:=FALSE;
 METSHAPE:=1;
 DRAWBUBBLES(BUBLX,BUBLY); (*display bubbles*)
 REQUEST;    (*prompt for space bar to test gas*)
 BUBCYCLE(200);
 REQUEST;     (* erase prompt for space bar *)
 SPACEPR:=FALSE;
 DRAWMATCH(MATCHX,MATCHY,WHITE2); (* draw match *)
 REPEAT
    MOVEBUBBLES(BUBLX,BUBLY);
    MOVEMATCH(MATCHX,MATCHY);
    DELAY(SPEED DIV 2);
 UNTIL MATCHX>=BUBLX;
 DRAWMATCH(MATCHX,MATCHY,BLACK2); (*erase match*)
 SHOWEXPLOSION(METALX);
 IF KEYIN THEN CHECKKEY(SPACEPR);
 IF QUIT THEN EXIT(REACTION);
 BUBCYCLE(12);
 SWAPMETAL(METSHAPE);
 DY:=3; (*decrease height risen by bubbles in each cycle *)
 BUBCYCLE(25);
 SWAPMETAL(METSHAPE);
```

```
      BUBCYCLE(20);
      DRAWBLOCK(METAL[METSHAPE],2,0,0,16,6,METALX,METALY,MODE);(*erase*)
      DRAWBUBBLES(BUBLX,BUBLY);   (*erase bubbles *)
    END; (* CALCIUMREACTION *)


BEGIN (* REACTION *)
   SPACEPR:=FALSE;
   CASE MET OF
   NA: SODIUMREACTION;
   CA: CALCIUMREACTION;
   END; (*CASE*)
   IF NOT QUIT THEN GAS;
   IF NOT QUIT THEN LITMUSTEST;
END; (* REACTION *)

(*==============================================================*)
PROCEDURE CONCLUSION;
(*==============================================================*)
CONST WIDTH=8;

   (*-----------------------------------------------------------*)
   PROCEDURE BIGSTAT(Y:INTEGER; S:STRING);
   (*-----------------------------------------------------------*)
   VAR LETTER,NUM,X :INTEGER; CH:CHAR;
   BEGIN
     X:=(XMAX-(10*LENGTH(S))) DIV 2;
     FOR LETTER:=1 TO LENGTH(S) DO
       BEGIN
         MOVETO(X,Y);
         NUM:=ORD(S[LETTER]);
         IF NUM IN [65..90] THEN NUM:=NUM-65  (* A..Z *)
              ELSE IF NUM=43 THEN NUM:=26; (* + *)
         WCHAR(CHR(NUM));
         X:=X+10;
       END;
   END; (* BIGSTAT *)

   (*-----------------------------------------------------------*)
   PROCEDURE ENCLOSE(WIDTH:INTEGER; COL:SCREENCOLOR);
   (*-----------------------------------------------------------*)
   BEGIN
     FILLAREA(XMIN,XMIN+WIDTH,YMIN,YMAX,COL);
     FILLAREA(XMAX-WIDTH,XMAX,YMIN,YMAX,COL);
     FILLAREA(XMIN,XMAX,YMIN,YMIN+WIDTH,COL);
     FILLAREA(XMIN,XMAX,YMAX-WIDTH,YMAX,COL);
   END; (* ENCLOSE *)

   (*-----------------------------------------------------------*)
   PROCEDURE DOWNARROW(X,Y1,Y2:INTEGER; COL:SCREENCOLOR);
   (*-----------------------------------------------------------*)
   CONST WID=3;
   VAR X1,X2:INTEGER;
   BEGIN
     FILLAREA(X-WID,X+WID,Y2,Y1,COL);
     X1:=X-3*WID;
```

```
 X2:=X+3*WID;
 REPEAT
  DRAWLINE(X1,Y2,X2,Y2,COL);
  Y2:=Y2-1;
  X1:=X1+1;
  X2:=X2-1;
 UNTIL X1>=X2;
END; (* DOWNARROW *)

(*------------------------------------------------------------------*)
PROCEDURE TEXT;
(*------------------------------------------------------------------*)
TYPE METAL=(NA,LI,CA);
VAR BAND:INTEGER;
   AMETAL:METAL;

  (*.................................................................*)
  PROCEDURE EQUATION;
  (*.................................................................*)
  VAR Y:INTEGER; CH:CHAR;
  BEGIN
   Y:=YMAX-30;
   BIGSTAT(Y,'WATER'); Y:=Y-15;
   BIGSTAT(Y,'+'); Y:=Y-15;
   BIGSTAT(Y,'VERY ACTIVE METAL'); Y:=Y-15;
   DOWNARROW(XMAX DIV 2,Y,Y-20,WHITE);  Y:=Y-55;
   BIGSTAT(Y,'HYDROGEN'); Y:=Y-15;
   BIGSTAT(Y,'+'); Y:=Y-15;
   BIGSTAT(Y,'METAL HYDROXIDE');
   GETKEY(CH,[SPACE,'Q']);
   IF QUIT THEN EXIT(TEXT);
  END; (* EQUATION *)

  (*.................................................................*)
  PROCEDURE CHANGE(MET:METAL);
  (*.................................................................*)
  VAR Y:INTEGER; CH:CHAR;
     S1,S2,METSTR:STRING[10];

        PROCEDURE BLANKLINE(Y,X1,X2:INTEGER);
        CONST BLANK='          ';
        BEGIN
         WHILE X1<X2 DO
          BEGIN
           WSTAT(X1,Y,BLANK); X1:=X1+70
          END;
        END; (* BLANKLINE *)

  BEGIN (* CHANGE *)
    CASE MET OF
      NA: METSTR:='SODIUM';
      LI: METSTR:='LITHIUM';
      CA: METSTR:='CALCIUM';
      END; (*CASE*)
    Y:=YMAX-60;
    BLANKLINE(Y,XMIN+20,XMAX-60);
```

```
            BIGSTAT(Y,METSTR);
            Y:=Y-100;
            BLANKLINE(Y,XMIN+20,XMAX-60);
            BIGSTAT(Y,CONCAT(METSTR,' HYDROXIDE'));
            GETKEY(CH,[SPACE,'Q']);
            IF QUIT THEN EXIT(TEXT);
          END; (* CHANGE *)

        BEGIN
         EQUATION;
         FOR AMETAL:=NA TO CA DO CHANGE(AMETAL);
        END; (* TEXT *)


      BEGIN (* CONCLUSION *)
       CHARTYPE(10);
       INITTURTLE;
       ENCLOSE(WIDTH,VIOLET);
       TEXT;
       CHARTYPE(6)
      END; (* CONCLUSION *)



BEGIN (* MACRO *)
  INITVARS;
  REPEAT
    FOR MET:=NA TO CA DO
     IF NOT QUIT THEN
      BEGIN
        INITTURTLE;
        DRAWBEAKER(BEAKERX,BEAKERY,BSIZE,WHITE2);
        FILLBEAKER(BEAKERX,LEVEL,BSIZE);
        METALX:=BEAKERX+46;   METALY:=178;
        DROPMETAL(METALX,METALY,MET);
        IF NOT QUIT THEN REACTION;
      END;
    IF NOT QUIT THEN CONCLUSION;
  UNTIL FIN('MACRO');
END; (* MACRO *)

(*******************************************************)
PROCEDURE SELECT(VAR CH: CHAR);
(*******************************************************)
CONST DEMO='SCOPIC demonstration     ......(';
VAR X,Y: INTEGER;
BEGIN
   TEXTMODE;
   PAGE(OUTPUT);
   X:=0; Y:=1;
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
   WRITE(AT(8,Y),'REACTION BETWEEN A VERY'); Y:=Y+1;
   WRITE(AT(9,Y),'ACTIVE METAL & WATER'); Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+4;;
   WRITE(AT(X,Y),'MACRO',DEMO,'1)'); Y:=Y+3;
   WRITE(AT(X,Y),'MICRO',DEMO,'2)'); Y:=Y+3;
```

```
    WRITE(AT(X,Y),'QUIT - back to main menu      ......(Q)'); Y:=Y+3;
    WRITE(AT(X+10,Y),'Select option  ..........(    )');
      GETTEXTCHAR(37,Y,CH,['1','2','Q']);
     QUIT:=CH='Q';
    PAGE(OUTPUT);
END; (* SELECT *)

(**************************************************************)
PROCEDURE BACKTOMENU;
(**************************************************************)
BEGIN
    PAGE(OUTPUT);
    WRITE(AT(10,8),'R E L O A D I N G');
    WRITE(AT(10,11),'M A I N   M E N U  . . . . . . . . . . .');
     SETCHAIN('  :DEMOMENU');
END;  (* BACKTOMENU *)

BEGIN  (* MAIN *)
     FALSEARRAY(BLANK);
     INITWATER;
     INITMETAL;
     CHARTYPE(MODE);
    REPEAT
       SELECT(OPTION);
      IF NOT QUIT THEN
        BEGIN
          CASE OPTION OF
             '1' : MACRO;
             '2' : MICRO;
             END; (*CASE*)
           QUIT:=FALSE;
        END;
     UNTIL QUIT;
     BACKTOMENU;
 END. (*ACTIVEMETAL*)
```

```
(*$S++*)
PROGRAM CARBONATE;
USES TURTLEGRAPHICS,CHAINSTUFF,USEFUL;

CONST  MODE=6;
TYPE
  BIGSHAPE=PACKED ARRAY[1..32,1..32] OF BOOLEAN;
  PH=(NEUTRAL,ACIDIC,BASIC);
  MOLECULE= RECORD
              X,Y,DX,DY : INTEGER;
              SHAPE :BIGSHAPE;
            END;
VAR
   QUIT : BOOLEAN;
   OPTION :CHAR;                                        (*  for MACRO or MICRO    *)
   BLANK,
   CARBATE,                                             (* shape of carbonate ion *)
   ATOM :BIGSHAPE;                                      (* shape of chloride ion  *)
   ACIDMOL : ARRAY[1..2] OF MOLECULE;                   (* shape of 2 H30+ ions *)
   WATERMOL :ARRAY[1..2] OF  MOLECULE;                  (* shape of 2 H20 molecules*)
   CO2:  PACKED ARRAY [1..8,1..32] OF BOOLEAN;          (*  shape of CO2  *)
   HATOM :PACKED ARRAY[1..5,1..5]OF BOOLEAN;            (*  shape of H+   *)
   HYDROGEN :PACKED ARRAY[1..5,1..16]OF BOOLEAN;        (*  shape of H2   *)

(***************************************************************)
PROCEDURE GETKEY(VAR ACH :CHAR; LEGALSET :CHARSET);
(***************************************************************)
BEGIN
 GETACHAR(ACH,LEGALSET);
 QUIT :=(ACH='Q');
END;

(***************************************************************)
FUNCTION FIN(ST :STRING) :BOOLEAN;
(***************************************************************)
VAR CH :CHAR;
BEGIN
  INITTURTLE;
  WSTAT(30,100,CONCAT('REPEAT ',ST));
  WSTAT(30,70,'DEMONSTRATION? (Y/N)');
  CHARTYPE(10);
  GETHICHAR(165,70,CH,['Y','N','Q']);
  CHARTYPE(MODE);
  FIN :=CH<>'Y';
  QUIT :=((CH='Q') OR (CH='q'));
END; (* FIN *)

(***************************************************************)
PROCEDURE FALSEARRAY(VAR NEWARRAY :BIGSHAPE);
(***************************************************************)
CONST MAX=32;
VAR  ROW,COL : INTEGER;
BEGIN
 FOR ROW :=1 TO MAX DO  FOR COL :=1 TO MAX DO  NEWARRAY[ROW,COL] :=FALSE;
END; (* FALSEARRAY *)
```

```
(*************************************************************)
PROCEDURE DEFINESHAPE(VAR NEWARRAY:BIGSHAPE; ACIDITY:PH; ANUM:INTEGER);
(*************************************************************)
(* Defines shape of acid & water molecules 2 orientations (1 & 2 determined by 'anum')
of each molecule is available. The array, 'NEWARRAY' must be initialized to ALL FALSE!*)
VAR CH:CHAR;

   (*===========================================================*)
    PROCEDURE INIT(WIDTH,HEIGHT:INTEGER; SYMBOL:CHAR);
   (*===========================================================*)
   VAR MAXCOL:INTEGER;

      (*--------------------------------------------------------*)
      PROCEDURE MERGE(ROW:INTEGER; S:STRING);
      (*--------------------------------------------------------*)
      VAR  COL: INTEGER;
      BEGIN
       FOR COL:=1 TO MAXCOL DO
          NEWARRAY[ROW+HEIGHT,COL+WIDTH]:=(S[COL]='X');
      END; (* MERGE *)

    BEGIN
     CASE SYMBOL OF
     'H': BEGIN
           MAXCOL:=5;
           MERGE(5, 'X   X');
           MERGE(4, 'X   X');
           MERGE(3, 'XXXXX');
           MERGE(2, 'X   X');
           MERGE(1, 'X   X');
         END; (* INITH *)
      'O':BEGIN
           MAXCOL:=9;
           MERGE(8, '  XXXXX  ');
           MERGE(7, ' X     X ');
           MERGE(6, 'X       X');
           MERGE(5, 'X       X');
           MERGE(4, 'X       X');
           MERGE(3, 'X       X');
           MERGE(2, ' X     X ');
           MERGE(1, '  XXXXX  ');
         END;
      'C':BEGIN
           MAXCOL:=8;
           MERGE(8, '  XXXXX ');
           MERGE(7, ' X     X');
           MERGE(6, 'X       ');
           MERGE(5, 'X       ');
           MERGE(4, 'X       ');
           MERGE(3, 'X       ');
           MERGE(2, ' X     X');
           MERGE(1, '  XXXXX ');
         END; (* INITC *)
      '+': BEGIN
           MAXCOL:=5;
           MERGE(5, '  X  ');
           MERGE(4, '  X  ');
           MERGE(3, 'XXXXX');
```

```
            MERGE(2,'   X   ');
            MERGE(1,'   X   ');
          END;
      '1': BEGIN
            MAXCOL:=4;
            MERGE(1,'XXXX');
          END;
      '2': BEGIN
            MAXCOL:=3;
            MERGE(3,'   X');
            MERGE(2,' X  ');
            MERGE(1,'X   ');
          END;
      '3': BEGIN
            MAXCOL:=3;
            MERGE(3,'X   ');
            MERGE(2,' X  ');
            MERGE(1,'   X');
          END;
       END;(*CASE*)
     END; (* INIT *)


BEGIN (* DEFINESHAPE *)
  CASE ACIDITY OF
    ACIDIC: CASE ANUM OF
        1 :BEGIN
             INIT(12,12,'0');
             INIT(27,0,'H'); (*this is proton removed in reaction*)
             INIT(27,27,'H'); INIT(0,13,'H');
             INIT(27,14,'+');
             INIT(6,15,'1'); INIT(22,23,'2');
             INIT(22,6,'3'); (*this is bond broken in reaction *)
           END;
        2:BEGIN
             INIT(9,12,'0');
             INIT(0,0,'H');(*this is proton donated in reaction*)
             INIT(0,27,'H'); INIT(27,13,'H');
             INIT(0,14,'+');
             INIT(22,15,'1'); INIT(6,6,'2'); (*bond broken*)
             INIT(6,22,'3');
           END;
          END;(*ACIDIC*)
    NEUTRAL: CASE ANUM OF
         1 :BEGIN (* all y ordinates will be 10 less than in corresponding acid shape *)
              INIT(12,2,'0');
              INIT(0,3,'H'); INIT(27,17,'H');
              INIT(6,5,'1'); INIT(22,13,'2');
            END;
         2:BEGIN
              INIT(9,2,'0');
              INIT(0,17,'H'); INIT(27,3,'H');
              INIT(22,5,'1'); INIT(6,12,'3');
            END;
         3: BEGIN
             INIT(14,8,'0');
             INIT(0,0,'H');
```

```
            INIT(0,16,'H');
            INIT(8,4,'2');
            INIT(8,15,'3');
           END;
         4: BEGIN
             INIT(0,8,'0');
             INIT(16,0,'H');
             INIT(16,16,'H');
             INIT(10,15,'2');
             INIT(10,4,'3');
           END;
         5:BEGIN
             INIT(8,0,'0');
             INIT(0,16,'H');
             INIT(18,16,'H');
             INIT(14,10,'2');
             INIT(7,10,'3');
           END;
         6:BEGIN
             INIT(8,16,'0');
             INIT(0,3,'H');
             INIT(16,3,'H');
             INIT(5,9,'2');
             INIT(14,9,'3');
           END;
         7:BEGIN (* CARBONATE *)
             INIT(15,13,'C');
             INIT(23,0,'0'); (*this is proton removed *)
             INIT(23,24,'0');  INIT(0,13,'0');
             INIT(27,15,'1');(*neg charge*)
             INIT(27,18,'1');(*neg charge*)
             INIT(10,17,'1'); INIT(20,22,'2');
             INIT(20,8,'3'); (* this is bond broken *)
           END;
        END; (* CASE OF NEUTRAL *)
   END; (*CASEOF ACIDITY*)
END; (* DEFINESHAPE *)

(**********************************************************)
PROCEDURE INITACARB;
(**********************************************************)
VAR  J: INTEGER;
BEGIN
 FOR J:=1 TO 2 DO
   BEGIN
    WITH ACIDMOL[J] DO BEGIN SHAPE:=BLANK; DEFINESHAPE(SHAPE,ACIDIC,J); END;
  WITH WATERMOL[J] DO BEGIN SHAPE:=BLANK; DEFINESHAPE(SHAPE,NEUTRAL,J); END;
   END; (* FOR *)
  CARBATE:=BLANK;
  DEFINESHAPE(CARBATE,NEUTRAL,7);
END; (* INITA&CARB *)
```

```
(**************************************************************)
PROCEDURE INITSHAPES;
(**************************************************************)

  (*============================================================*)
  PROCEDURE INITCO2;
  (*============================================================*)

    (*----------------------------------------------------------*)
    PROCEDURE INIT(ROW:INTEGER; S:STRING);
    (*----------------------------------------------------------*)
    VAR COL : INTEGER;
    BEGIN
     FOR COL:=1 TO 32 DO  CO2[ROW,COL]:=S[COL]='X';
    END; (* INIT *)

  BEGIN
    INIT(8,'  XXX         XXXX         XXX  ');
    INIT(7,' X   X       X    X       X   X ');
    INIT(6,'X     X     X            X     X');
    INIT(5,'X     X XXX X       XXXX X     X');
    INIT(4,'X     X     X            X     X');
    INIT(3,'X     X XXX X       XXXX X     X');
    INIT(2,' X   X       X    X       X   X ');
    INIT(1,'  XXX         XXXX         XXX  ');
  END; (* INITCO2 *)

  (*============================================================*)
  PROCEDURE INITATOM;
  (*============================================================*)
  VAR   STR: ARRAY[1..32] OF STRING;   J:INTEGER;

    (*----------------------------------------------------------*)
    PROCEDURE INIT(ROW:INTEGER; VAR BITS:BIGSHAPE; S:STRING);
    (*----------------------------------------------------------*)
    VAR  COL: INTEGER;
    BEGIN
     FOR COL:=1 TO 32 DO   BITS[ROW,COL]:=S[COL]='X';
    END;  (* INIT*)

  BEGIN
      STR[1]:='           XXXXXXXX             ';
      STR[2]:='        XXXXXXXXXXXXXXX         ';
      STR[3]:='      XXXXXXXXXXXXXXXXXXX       ';
      STR[4]:='     XXXXXXXXXXXXXXXXXXXXX      ';
      STR[5]:='    XXXXXXXXXXXXXXXXXXXXXXX     ';
      STR[6]:='   XXXXXXXXXXXXXXXXXXXXXXXXX    ';
      STR[7]:='   XXXXXXXXXXXXXXXXXXXXXXXXXX   ';
      STR[8]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXX   ';
      STR[9]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
     STR[10]:='  XXXXXXXXXXXXXXXXXXXXXXXXXXXX  ';
     STR[11]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
     STR[12]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
     STR[13]:=' XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ';
     STR[14]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
     STR[15]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
     STR[16]:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
   FOR J:=1 TO 16 DO
```

```
    BEGIN
      INIT(J,ATOM,STR[J]);
      INIT(33-J,ATOM,STR[J]);
    END;
  END; (* INITATOM *)

BEGIN (* INITSHAPES *)
  INITCO2;
  INITACARB;
  INITATOM;
END; (*INITSHAPES*)

(**********************************************************)
PROCEDURE MICRO;
(**********************************************************)
CONST  WIDTH=55;  (* Total width of each carbonate made up of:
                       cation(16) + space (4) + carbonate ion(32) + space (3) *)
TYPE  MEDSHAPE=PACKED ARRAY[1..16,1..16] OF BOOLEAN;
      ION=RECORD
             X,Y,DX,DY: INTEGER;
          END;
      IONTYPE=(CA,NA,MET);
VAR METION:IONTYPE;  CAION : ION;
     BLANKION,CATION:MEDSHAPE;
     CH: CHAR;  NUM : INTEGER;
     BASELEVEL, CARBX,CARBY:INTEGER;      (* coord. carbonate ion *)

  (*============================================================*)
  PROCEDURE DRAWARROW(X,Y,SIZE:INTEGER);
  (*============================================================*)
  VAR TIP: INTEGER;
  BEGIN
    TIP:=7;
    MOVECOL(X,Y,WHITE1);
    MOVECOL(X+SIZE,Y,NONE);
    MOVECOL(X+SIZE-TIP,Y+TIP,WHITE1);
    MOVETO(X+SIZE,Y);
    MOVECOL(X+SIZE-TIP,Y-TIP,NONE);
  END; (* DRAWARROW *)

  (*============================================================*)
  PROCEDURE MERGEBITS(ROW:INTEGER;VAR BITS:MEDSHAPE; S:STRING);
  (*============================================================*)
  VAR  COL:INTEGER;
  BEGIN
   FOR COL:=1 TO 16 DO  BITS[ROW,COL]:=S[COL]='X';
  END; (*MERGEBITS*)

  (*============================================================*)
  PROCEDURE INITION(ANION:IONTYPE; VAR ANYION:MEDSHAPE);
  (*============================================================*)
  BEGIN
   ANYION:=BLANKION;
   CASE ANION OF
```

```
  NA:BEGIN
       MERGEBITS(15,ANYION,'   XXXXXX  X    ');
       MERGEBITS(14,ANYION,'  XXXXXXX  XXX  ');
       MERGEBITS(13,ANYION,' XXXXXXX     XX ');
       MERGEBITS(12,ANYION,' XXXXXXXX  XXXX ');
       MERGEBITS(11,ANYION,'XXXXXXXXX  XXXXX');
       MERGEBITS(10,ANYION,'XX  XX  XXXXXXXX');
       MERGEBITS(9,ANYION,'XX   X  XXXXXXXX');
       MERGEBITS(8,ANYION,'XX      XX    XX');
       MERGEBITS(7,ANYION,'XX  XX  X  X  XX');
       MERGEBITS(6,ANYION,' X  XX  XX    X ');
     END;
  CA:BEGIN
       MERGEBITS(10,ANYION,'XXX     XXXXXXXX');
       MERGEBITS(9,ANYION,'XX  XXXXXXXXXXXX');
       MERGEBITS(8,ANYION,'XX  XXXXX     XX');
       MERGEBITS(7,ANYION,'XX  XXXX  XX  XX');
       MERGEBITS(6,ANYION,' XX    XX     X ');
     END;
  MET:BEGIN
       MERGEBITS(9,ANYION,'XXXX   XX   XXXX');
       MERGEBITS(8,ANYION,'XXXX        XXXX');
       MERGEBITS(7,ANYION,'XXXX  X  X  XXXX');
       MERGEBITS(6,ANYION,'XXXX  X  X  XXXX');
       MERGEBITS(5,ANYION,' XXX  XXXX  XXX ');
     END;
   END;(*CASE*)
  END; (* INITION *)

(*==============================================================*)
PROCEDURE INITMICROSHAPES;
(*==============================================================*)
  (*------------------------------------------------------------*)
  PROCEDURE INITBLANK(VAR ANYION:MEDSHAPE);
  (*------------------------------------------------------------*)
  BEGIN
    MERGEBITS(16,ANYION,'      XXXX      ');
    MERGEBITS(15,ANYION,'   X  XXX  X    ');
    MERGEBITS(14,ANYION,'  XX  XXX  XXX  ');
    MERGEBITS(13,ANYION,' X     X     XX ');
    MERGEBITS(12,ANYION,' XXX  XXX  XXXX ');
    MERGEBITS(11,ANYION,'XXXX  XXX  XXXXX');
    MERGEBITS(10,ANYION,'XXXXXXXXXXXXXXXX');
     MERGEBITS(9,ANYION,'XXXXXXXXXXXXXXXX');
     MERGEBITS(8,ANYION,'XXXXXXXXXXXXXXXX');
     MERGEBITS(7,ANYION,'XXXXXXXXXXXXXXXX');
     MERGEBITS(6,ANYION,'XXXXXXXXXXXXXXXX');
     MERGEBITS(5,ANYION,' XXXXXXXXXXXXXX ');
     MERGEBITS(4,ANYION,'  XXXXXXXXXXXX  ');
     MERGEBITS(3,ANYION,'  XXXXXXXXXXXX  ');
     MERGEBITS(2,ANYION,'   XXXXXXXXX    ');
     MERGEBITS(1,ANYION,'     XXXXXX     ');
  END; (* INITBLANK *)

  (*------------------------------------------------------------*)
  PROCEDURE INITHATOM;
  (*------------------------------------------------------------*)
```

```
      (*..................................................................................*)
      PROCEDURE INIT(ROW:INTEGER; S:STRING);
      (*..................................................................................*)
      VAR COL: INTEGER;
      BEGIN
       FOR COL:=1 TO 5 DO  HATOM[ROW,COL]:=S[COL]='X';
      END; (* INIT *)

    BEGIN
      INIT(5,'X   X');
      INIT(4,'X   X');
      INIT(3,'XXXXX');
      INIT(2,'X   X');
      INIT(1,'X   X');
    END; (* INITHATOM *)


  BEGIN (* INITMICROSHAPES *)
   INITBLANK(BLANKION);
   INITION(CA,CATION);
   INITHATOM;
  END; (* INITMICROSHAPES *)


(*$I :CARB1 *)
(*$I :CARB2*)
(*$I :CARB3*)
(*$I :CARB4*)
(*$I :CARB5*)
(*$I :CARB6*)
```

```
(* CARB1 to be included in CARBONATE *)
  (*=======================================================================*)
  PROCEDURE EXPLAINSHAPES;
  (*=======================================================================*)
  TYPE BIGSHAPE= PACKED ARRAY[1..18,1..18] OF BOOLEAN;
  VAR S1,S2:STRING;  CH:CHAR;
      X,Y: INTEGER;
      BIGMG: BIGSHAPE;

    (*---------------------------------------------------------------------*)
    PROCEDURE SHOWTEXT;
    (*---------------------------------------------------------------------*)
    BEGIN
     INITTURTLE;
     WSTAT(0,172,S1);
     WSTAT(0,152,S2);
    END;
```

```
BEGIN (* EXPLAINSHAPES *)
  S1 :='This demonstration will display';
  S2 :='the following structures :-';
  SHOWTEXT;  X :=140; Y :=100;
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X,Y,MODE);
  WSTAT(XMIN,Y,'WATER MOLECULE :'); Y :=55;
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X,Y-10,MODE);
  WSTAT(XMIN,Y,'HYDRONIUM ION :'); Y :=5;
  DRAWBLOCK(CO2,4,0,0,32,8,X,Y,MODE);
  WSTAT(XMIN,Y,'CARBON DIOXIDE :');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(MICRO);
  S1 :='Other structures displayed';
  S2 :='in this demonstration :-';
  SHOWTEXT;  Y :=110;
  DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
  WSTAT(XMIN,Y,'CALCIUM ION :');
  INITION(NA,CATION);  Y :=60;
  DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
  WSTAT(XMIN,Y,'SODIUM ION :');   Y :=5;
  DRAWBLOCK(CARBATE,4,0,0,32,32,X,Y,MODE);
  WSTAT(XMIN,Y+8,'CARBONATE ION :');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(MICRO);
END; (* EXPLAINSHAPES *)

(*================================================================*)
PROCEDURE SOLVATECATIONS;
(*================================================================*)
CONST WATERNUM=4;
VAR WATER: ARRAY[1..WATERNUM] OF BIGSHAPE;  AQUA :INTEGER;

  (*--------------------------------------------------------------*)
  PROCEDURE DEFINEWATER;
  (*--------------------------------------------------------------*)
  (*Watermolecule shapes numbered 1 - 4 correspond to  shapes 3-6 in
    Defineshape*)
  BEGIN (* DEFINEWATER *)
   WHILE ((AQUA<=WATERNUM) AND (NOT KEYIN)) DO
    BEGIN
      WATER[AQUA]:=BLANK;
      DEFINESHAPE(WATER[AQUA],NEUTRAL,AQUA+2);
      AQUA :=AQUA+1;
    END;
  END; (* DEFINEWATER *)

  (*--------------------------------------------------------------*)
  PROCEDURE HYDRATE(ANUM,CENTRX,CENTRY :INTEGER);
  (*--------------------------------------------------------------*)
  VAR   SIZE,DISTANCE,RADIUS,BONDLEN,
        X,Y,X1,Y1,X2,Y2,J :INTEGER;
  BEGIN
    SIZE :=24;  DISTANCE :=20;
    BONDLEN :=4; RADIUS :=15;
     FOR J :=1 TO ANUM DO
      BEGIN
```

```
      CASE J OF
      1: BEGIN
           X:=CENTRX-DISTANCE-SIZE;
           Y:=CENTRY-(SIZE DIV 2);
           X1:=CENTRX-RADIUS-2; Y1:=CENTRY;
           X2:=X1+BONDLEN; Y2:=Y1;
         END;
      2: BEGIN
           X:=CENTRX+DISTANCE;
           Y:=CENTRY-(SIZE DIV 2);
           X1:=CENTRX+RADIUS; Y1:=CENTRY;
           X2:=X1-BONDLEN; Y2:=Y1;
         END;
      3: BEGIN
           X:=CENTRX-(SIZE DIV 2);
           Y:=CENTRY+DISTANCE;
           X1:=CENTRX-2; Y1:=CENTRY+RADIUS;
           X2:=X1;     Y2:=Y1-BONDLEN;
         END;
      4: BEGIN
           X:=CENTRX-(SIZE DIV 2);
           Y:=CENTRY-DISTANCE-SIZE;
           X1:=CENTRX-2; Y1:=CENTRY-RADIUS;
           X2:=X1;      Y2:=Y1+BONDLEN;
         END;
        END; (*CASE*)
    DRAWBLOCK(WATER[J],4,0,0,24,24,X,Y,MODE);
    DRAWLINE(X1,Y1,X2,Y2,WHITE2);
  END;
END; (* HYDRATE *)

(*-------------------------------------------------------------------------*)
PROCEDURE SHOWHYDRATE;
(*-------------------------------------------------------------------------*)
CONST ST='Press <SPACE BAR> to show hydrated ion';
VAR X,Y, MIDX,MIDY : INTEGER; CH:CHAR;
BEGIN
   INITION(MET,CATION);
   INITTURTLE;   X:=10; Y:=YMAX-15;
   WSTAT(X,Y,'In aqueous solution ions are HYDRATED.');
   MIDX:=(XMAX DIV 2)-10; MIDY:=(YMAX DIV 2)+20;
   DRAWBLOCK(CATION,2,0,0,16,16,MIDX-8,MIDY-8,MODE);
   WSTAT(XMIN,YMIN+40,'This represents any metal ion.');
   REPEAT
      DEFINEWATER;  IF KEYIN THEN READ(CH);
   UNTIL AQUA>WATERNUM;
   WSTAT(XMIN,YMIN,ST);
   GETKEY(CH,[SPACE,'Q']);
   IF QUIT THEN EXIT(SOLVATECATIONS);
   WSTAT(XMIN,YMIN,ST);
   HYDRATE(WATERNUM,MIDX,MIDY);
   WSTAT(XMIN,15,'The number of water molecules involved');
   WSTAT(XMIN,YMIN,'in hydration varies for each ion.');
   GETKEY(CH,[SPACE,'Q']);
   IF QUIT THEN EXIT(SOLVATECATIONS);
END; (* SHOWHYDRATE *)
```

```
(*---------------------------------------------------------------------------*)
PROCEDURE NETMETALREACTION;
(*---------------------------------------------------------------------------*)
VAR X,Y : INTEGER; CH:CHAR;
  (*.........................................................................*)
  PROCEDURE LABELANIONS(XX,YY:INTEGER);
  (*.........................................................................*)
  VAR  LEN, RADIUS,X,Y,J  : INTEGER;
  BEGIN
   RADIUS:=45;   LEN:=7*LENGTH('anion');
   FOR J:=1 TO 4 DO
    BEGIN
     CASE J OF
      1: BEGIN    X:=XX-RADIUS; Y:=YY-3;          END;
      2: BEGIN    X:=XX+RADIUS-LEN; Y:=YY-3;     END;
      3: BEGIN    X:=XX-(LEN DIV 2); Y:=YY+12;   END;
      4: BEGIN    X:=XX-(LEN DIV 2); Y:=YY-18;   END;
      END; (*CASE*)
     WSTAT(X,Y,'anion');
    END; (*for*)
  END; (* LABELANIONS *)

 BEGIN (* NETMETALREACTION *)
   INITTURTLE;
   X:=0; Y:=YMAX-10;
   WSTAT(X,Y,'Net reaction of cation:-'); Y:=Y-40;
   WSTAT(X,Y,'CATION IN'); Y:=Y-10;
   WSTAT(X,Y,'CRYSTAL');DRAWARROW(X+90,Y+4,20);
   WSTAT(X+130,Y,'HYDRATED CATION'); Y:=Y-10;
   WSTAT(X,Y,'LATTICE'); Y:=Y-50; X:=X+50;
   DRAWBLOCK(CATION,2,0,0,16,16,X-8,Y-8,MODE);
   LABELANIONS(X,Y); X:=X+130;
   DRAWBLOCK(CATION,2,0,0,16,16,X-8,Y-8,MODE);
   HYDRATE(4,X,Y);
   GETKEY(CH,[SPACE,'Q']);
 END; (* NETMETALREACTION *)


BEGIN (* SOLVATECATION *)
  AQUA:=1;
  SHOWHYDRATE;
  NETMETALREACTION;
END; (* SOLVATECATIONS *)



(* CARB2 to be included with CARBONATE *)
  (*=========================================================================*)
  PROCEDURE REACTION;
  (*=========================================================================*)
  VAR  CYCLE,STEP : INTEGER;
       PROTONX,PROTONY :ARRAY[1..2] OF INTEGER;
```

```
(*-------------------------------------------------------------------*)
PROCEDURE CHECKKEY;
(*-------------------------------------------------------------------*)
VAR CH:CHAR;
BEGIN
 READ(CH);
 IF CH=SPACE THEN
    BEGIN
      CHARTYPE(10);
      WSTAT(186,182,'continue');
      GETKEY(CH,[SPACE,'Q']);
      WSTAT(186,182,'pause   ');
      CHARTYPE(MODE);
    END
   ELSE  QUIT:=((CH='Q') OR (CH='q'));
 IF QUIT THEN EXIT(REACTION);
END; (* CHECKKEY *)

(*-------------------------------------------------------------------*)
PROCEDURE MOVEMOLECULE(VAR ANYREC:MOLECULE;  HEIGHT:INTEGER);
(*-------------------------------------------------------------------*)
BEGIN
 WITH ANYREC DO
   BEGIN
     DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
     X:=X+DX;  Y:=Y+DY;
     DRAWBLOCK(SHAPE,4,0,0,32,HEIGHT,X,Y,MODE);
   END;
END; (* MOVEMOLECULE *)

(*-------------------------------------------------------------------*)
PROCEDURE SHOWACID(NUM:INTEGER);
(*-------------------------------------------------------------------*)
BEGIN
 WITH ACIDMOL[NUM] DO DRAWBLOCK(SHAPE,4,0,0,32,32,X,Y,MODE);
END; (* SHOWACID *)

(*-------------------------------------------------------------------*)
PROCEDURE DISPLAYPROMPT;
(*-------------------------------------------------------------------*)
VAR CH: CHAR;
   (*.................................................................*)
   PROCEDURE PROMPT;
   (*.................................................................*)
   BEGIN
     WSTAT(0,182,'<SPACE BAR> to react acid with carbonate');
   END;

 BEGIN (* DISPLAYPROMPT *)
   PROMPT; (*display*)
   GETKEY(CH,[SPACE,'Q']);
   IF QUIT THEN EXIT(REACTION);
   PROMPT; (*erase*)
   WSTAT(40,182,'Press <SPACE BAR> to pause');
 END; (* DISPLAYPROMPT *)
```

```
(*----------------------------------------------------------------------*)
PROCEDURE MOVEACID(NUM:INTEGER);
(*----------------------------------------------------------------------*)
CONST   INCR=7; (* required in NEWVALUE to calculate starting position  of acid*)
VAR  STEP: INTEGER;
BEGIN
 FOR STEP:=1 TO INCR DO
    BEGIN
     MOVEMOLECULE(ACIDMOL[NUM],32);
     DELAY(10);
     IF KEYIN THEN
        BEGIN
          CHECKKEY;  IF QUIT THEN EXIT(MOVEACID);
        END;
    END;
END; (* MOVEACID *)

(*----------------------------------------------------------------------*)
PROCEDURE MOVEWATER(NUM:INTEGER);
(*----------------------------------------------------------------------*)
BEGIN
   MOVEMOLECULE(WATERMOL[NUM],24);
END; (* MOVEWATER *)

(*----------------------------------------------------------------------*)
PROCEDURE DRAWCARBONATE(VAR ANYION:MEDSHAPE;  VAR SYMBOL:IONTYPE);
(*----------------------------------------------------------------------*)
VAR   J,LASTX,
       X,Y,           (* Co-ord. at which to draw ions *)
      HEIGHT:INTEGER; (*Y-spacing of cation above carbonate*)
      METALSTR:STRING[10];
BEGIN
  X:=XMIN; (*start drawing ions at L.H.S *)
  BASELEVEL:=0;
  HEIGHT:=8;
  LASTX:=XMAX-32;
  CASE SYMBOL OF
     CA: METALSTR:='calcium';
     NA: METALSTR:='sodium';
     END; (*CASE*)
  REPEAT
     DRAWBLOCK(CARBATE,4,0,0,32,32,X+20,BASELEVEL,MODE);
     IF SYMBOL=NA THEN
     DRAWBLOCK(ANYION,2,0,0,16,16,X+(WIDTH DIV 2),BASELEVEL+28,MODE);
     DRAWBLOCK(ANYION,2,0,0,16,16,X,BASELEVEL+HEIGHT,MODE);
     X:=X+WIDTH;
  UNTIL (X>=LASTX);
  WSTAT(0,184,CONCAT('The surface ions of ',METALSTR,' carbonate'));
  WSTAT(0,174,'are represented by the structure:-');
  GETKEY(CH,[SPACE,'Q']);
  IF QUIT THEN EXIT(REACTION);
  FILLAREA(0,XMAX,172,YMAX,BLACK1);
 END; (* DRAWMETAL *)
```

```
(*------------------------------------------------------------------------*)
PROCEDURE NEWVALUE(ANYNUM:INTEGER; SYMBOL:IONTYPE);
(*------------------------------------------------------------------------*)
CONST   INCR=7;
VAR CENTCARB, TOPCARB,   (*coord. of top centre of target carbonate*)
     ANUM:INTEGER;  (* determines which carbonate reacts *)
BEGIN
  CASE ANYNUM OF
   1: BEGIN
       ANUM:=2;
       WITH CAION DO   BEGIN   DX:=-6; DY:=10;   END;
       WITH ACIDMOL[1] DO  BEGIN   DX:=4;  DY:=-12; END;
       WITH ACIDMOL[2] DO BEGIN    DX:=-12; DY:=-12; END;
      END; (* 1  *)
   2: BEGIN
       ANUM:=3;
       WITH CAION DO   BEGIN   DX:=-10; DY:=8;   END;
       WITH ACIDMOL[1] DO   BEGIN   DX:=10; DY:=-12; END;
       WITH ACIDMOL[2] DO   BEGIN    DX:=-12; DY:=-12; END;
      END; (* 2 *)
   3: BEGIN
       ANUM:=4;
       WITH CAION DO BEGIN   DX:=-12;  DY:=6;   END;
       WITH ACIDMOL[1] DO BEGIN   DX:=6; DY:=-12; END;
       WITH ACIDMOL[2] DO BEGIN    DX:=-12; DY:=-12; END;
      END; (* 3  *)
   4: BEGIN
       ANUM:=1;
       WITH CAION DO    BEGIN   DX:=-2;  DY:=10;   END;
       WITH ACIDMOL[1] DO   BEGIN   DX:=6; DY:=-12; END;
       WITH ACIDMOL[2] DO    BEGIN    DX:=-14; DY:=-12; END;
      END; (* 4  *)
   5: BEGIN
       ANUM:=0;
       WITH CAION DO  BEGIN   DX:=0;  DY:=10;   END;
       WITH ACIDMOL[1] DO BEGIN   DX:=0; DY:=-12; END;
       WITH ACIDMOL[2] DO BEGIN    DX:=-14; DY:=-12; END;
     END; (* 5  *)
    END;(* CASE*)

    CAION.X:=WIDTH*ANUM;(* see DRAWCARBONATE*)
    CAION.Y:=BASELEVEL+8;
    IF SYMBOL=NA THEN
     BEGIN
       CAION.X:=CAION.X + WIDTH DIV 2;
       CAION.Y:=BASELEVEL + 28; (* See DRAWCARBONATE *)
     END;
    TOPCARB:=BASELEVEL+32;
    CARBX:=WIDTH*ANUM+20;  (* x- coord of carbonate ion *)
    CENTCARB:=CARBX +16;
    WITH ACIDMOL[1] DO
     BEGIN
       X:=CENTCARB-42; (*see DEFINESHAPE -proton removed
                    is 22 from bottom L.H.C.*)
       Y:=TOPCARB;
       X:=X- (INCR*DX);
```

```
     Y:=Y-(INCR*DY);
    END;
   WITH ACIDMOL[2] DO
    BEGIN
     X:=CENTCARB+10;
     Y:=TOPCARB+10;
     X:=X-(INCR*DX);
     Y:=Y-(INCR*DY);
    END;
 END; (* NEWVALUE *)

(*--------------------------------------------------------------------*)
PROCEDURE REACT(ANUM:INTEGER);
(*--------------------------------------------------------------------*)
VAR   I : INTEGER;
  (*..................................................................*)
  PROCEDURE POLARISE(NUM:INTEGER);
  (*..................................................................*)
  VAR X1,Y1 : INTEGER;
  BEGIN
    CHARTYPE(10);
     Y1:=12;
     CASE NUM OF
     1: X1:=26;
     2: X1:=0;
     END;(*CASE*)
     WITH ACIDMOL[NUM] DO
      BEGIN
        X1:=X+X1; Y1:=Y+Y1;
        WSTAT(X1,Y1,' '); (*erase + *)
        WSTAT(X1,Y1-6,'+');
      END;
    CHARTYPE(MODE);
    DELAY(20);
  END; (* POLARISE *)

  (*..................................................................*)
  PROCEDURE CALCULATE(NUM:INTEGER);
  (*..................................................................*)
  BEGIN
   WITH WATERMOL[NUM] DO
    BEGIN
      X:=ACIDMOL[NUM].X;
      Y:=ACIDMOL[NUM].Y+10;  (*see DEFINESHAPE *)
      DX:=-ACIDMOL[NUM].DX;
      DY:=-ACIDMOL[NUM].DY;
    END;
  END; (* CALCULATE *)

  (*..................................................................*)
  PROCEDURE SHOWWATER(NUM:INTEGER);
  (*..................................................................*)
  BEGIN
   WITH WATERMOL[NUM] DO  DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,MODE);
  END; (* SHOWWATER *)
```

```
(*..............................................................................*)
PROCEDURE REPLACEACID(NUM:INTEGER);
(*..............................................................................*)
BEGIN
  WITH ACIDMOL[NUM] DO  DRAWBLOCK(SHAPE,4,0,0,32,32,X,Y+6,0);
  SHOWWATER(ANUM);   (*display H20 *)
END; (* REPLACEACID *)

(*..............................................................................*)
PROCEDURE SHOWPROTON(X,Y:INTEGER);
(*..............................................................................*)
BEGIN
   DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
   WSTAT(X+2,Y+6,'+');
END; (* SHOWPROTON *)

(*..............................................................................*)
PROCEDURE ADDABOND(X,Y:INTEGER);
(*..............................................................................*)
BEGIN
  DRAWLINE(X+2,Y-2,X+2,Y-8,WHITE1);
  DELAY(50);
END; (* ADDABOND *)

(*..............................................................................*)
PROCEDURE REMOVECHARGE(NUM:INTEGER);
(*..............................................................................*)
VAR X,Y:INTEGER;
BEGIN
   IF NUM=1 THEN Y:=BASELEVEL+18 ELSE Y:=BASELEVEL+15;
   X:=CARBX+27;
   DRAWLINE(X-1,Y,X+2,Y,BLACK2);
END; (* REMOVECHARGE *)

BEGIN (* REACT *)
  POLARISE(ANUM); (*show +ve charge closer to carbonate*)
  CALCULATE(ANUM); (* coord & direction of H20  *)
  IF ANUM=1 THEN
     BEGIN
       PROTONX[1]:=ACIDMOL[1].X+27;(*see DEFINESHAPE*)
       PROTONY[1]:=ACIDMOL[1].Y;
     END
     ELSE
       BEGIN
         PROTONX[2]:=PROTONX[1] +25; (*20 space+5 width of H *)
         PROTONY[2]:=ACIDMOL[2].Y;
       END;
  IF KEYIN THEN CHECKKEY; (* wait then erase protons & electrons*)
  ADDABOND(PROTONX[ANUM],PROTONY[ANUM]);
  REPLACEACID(ANUM);
  REMOVECHARGE(ANUM);
  IF KEYIN THEN CHECKKEY;
  FOR I:=1 TO 8 DO
   BEGIN          (* move water *)
     MOVEWATER(ANUM); IF KEYIN THEN CHECKKEY;
   END;
```

```
 SHOWWATER(ANUM);   (* erase water *)
END; (* REACT *)

(*-----------------------------------------------------------------------*)
PROCEDURE COMPLETE(VAR SYMBOL :IONTYPE);
(*-----------------------------------------------------------------------*)
VAR  I,CO2X,CO2Y, TEMPX,TEMPY :INTEGER;
  (*.....................................................................*)
  PROCEDURE MOVECAION;
  (*.....................................................................*)
  VAR I:INTEGER;
  BEGIN
   WITH CAION DO
    BEGIN
     FOR I:=1 TO 12 DO
       BEGIN              (* erase *)
         DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
         X:=X+DX; Y:=Y+DY;
         DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);
         DELAY(6);              (*display *)
          IF KEYIN THEN CHECKKEY;
       END;
     DRAWBLOCK(CATION,2,0,0,16,16,X,Y,MODE);   (*erase cation*)
    END; (*WITH*)
  END;  (* MOVECAION *)

  (*.....................................................................*)
  PROCEDURE MOVECO2(VAR X,Y :INTEGER);
  (*.....................................................................*)
  BEGIN
    DRAWBLOCK(CO2,4,0,0,32,8,X,Y,MODE);
    Y:=Y+8;
    DRAWBLOCK(CO2,4,0,0,32,8,X,Y,MODE);
  END; (* MOVECO2 *)

  (*.....................................................................*)
  PROCEDURE DRAWPRODUCTS(X1 :INTEGER);
  (*.....................................................................*)
  BEGIN                    (* erase carbonic acid molecule *)
   FILLAREA(X1-5,X1+32,BASELEVEL,BASELEVEL+48,BLACK1);
   CO2X:=X1; CO2Y:=10;
   DRAWBLOCK(CO2,4,0,0,32,8,CO2X,CO2Y,MODE);
   WITH WATERMOL[1] DO      (*display CO2*)
    BEGIN
      X:=X1+10;    Y:=25;
      DX:=-DX;
      DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,MODE);
    END;               (*displayH20*)
  END; (* DRAWPRODUCTS *)

 BEGIN (*COMPLETE*)
  TEMPX:=CAION.X-(WIDTH DIV 2);
  TEMPY:=8; (*baselevel+height*)
  MOVECAION;
  DRAWPRODUCTS(PROTONX[1]);
  FOR I:=1 TO 5 DO
```

```
        BEGIN    (* Wait a couple of seconds before moving products*)
          DELAY(20);    IF KEYIN THEN CHECKKEY;
        END;
      FOR I:=1 TO 14 DO
        BEGIN
          MOVEWATER(1); (* move H20 *)
          MOVECO2(CO2X,CO2Y); (* move CO2 *)
           IF KEYIN THEN CHECKKEY;
        END;
      DRAWBLOCK(CO2,4,0,0,32,8,CO2X,CO2Y,MODE); (*erase CO2 *)
      WITH WATERMOL[1] DO DRAWBLOCK(SHAPE,4,0,0,32,24,X,Y,MODE);
       IF SYMBOL=NA THEN                                          (*erase H20*)
         BEGIN  (* move 2nd Na ion *)
           CAION.X:=TEMPX;
           CAION.Y:=TEMPY;
           MOVECAION;
         END;
    END; (* COMPLETE *)

BEGIN (* REACTION *)
  FOR METION:=CA TO NA DO
    IF NOT QUIT THEN
      BEGIN
        INITION(METION,CATION);
        INITTURTLE;
        DRAWCARBONATE(CATION,METION);
        CYCLE:=0;
        REPEAT
          CYCLE:=CYCLE+1;
          NEWVALUE(CYCLE,METION);
          IF CYCLE=1 THEN
            BEGIN
              SHOWACID(1); (* display acid molecule *)
              DISPLAYPROMPT;
              SHOWACID(1); (* erase acid molecule *)
            END;
          FOR STEP:=1 TO 2 DO
            BEGIN
              SHOWACID(STEP);
              MOVEACID(STEP);
               IF NOT QUIT THEN REACT(STEP);
            END;
         IF NOT QUIT THEN COMPLETE(METION);
        UNTIL ((CYCLE=4) OR QUIT);
      END; (*IF*)
 END; (* REACTION *)
```

```
(* CARB3 *)
  (*==============================================================*)
  PROCEDURE CONCLUSION;
  (*==============================================================*)

    (*------------------------------------------------------------*)
    PROCEDURE GETSPACE;
    (*------------------------------------------------------------*)
    VAR CH:CHAR;
    BEGIN
      GETKEY(CH,[SPACE,'Q']);
      IF QUIT THEN EXIT(CONCLUSION);
    END; (* GETSPACE *)

    (*------------------------------------------------------------*)
    PROCEDURE DRAWPLUS(X,Y:INTEGER);
    (*------------------------------------------------------------*)
    CONST  SIZE=10;
    BEGIN
      MOVECOL(X,Y,WHITE1);
      MOVECOL(X+SIZE,Y,NONE);
      X:=X+(SIZE DIV 2);
      Y:=Y+(SIZE DIV 2);
      MOVECOL(X,Y,WHITE1);
      MOVECOL(X,Y-SIZE,NONE);
    END; (* DRAWPLUS *)

    (*------------------------------------------------------------*)
    PROCEDURE DOWNARROW(X,Y,SIZE:INTEGER);
    (*------------------------------------------------------------*)
    CONST TIP=7;
    BEGIN
      MOVECOL(X,Y,WHITE2);
      MOVECOL(X,Y-SIZE,NONE);
      MOVECOL(X-TIP,Y-SIZE+TIP,WHITE1);
      MOVETO(X,Y-SIZE);
      MOVECOL(X+TIP,Y-SIZE+TIP,NONE);
    END; (* DOWNARROW *)

    (*------------------------------------------------------------*)
    PROCEDURE DRAWH2(X,Y:INTEGER);
    (*------------------------------------------------------------*)
    BEGIN
      DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
      DRAWBLOCK(HATOM,2,0,0,5,5,X,Y+15,MODE);
      MOVECOL(X+2,Y+7,WHITE1);
      MOVECOL(X+2,Y+12,NONE);
    END; (* DRAWH2 *)

    (*------------------------------------------------------------*)
    PROCEDURE NETHYDROGEN;
    (*------------------------------------------------------------*)
    VAR X,Y : INTEGER;
    BEGIN
      X:=1; Y:=80;
      INITTURTLE;
```

```
 WSTAT(XMIN,YMAX-10,'Net reaction of hydronium ions:-');
 DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X,Y-22,MODE);
 DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X,Y+22,MODE);
 X:=X+60;
 DRAWARROW(X,Y+16,30);
 X:=X+80;
 DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X,Y+22,MODE);
 DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X,Y-22,MODE);
 X:=X+60;
 DRAWPLUS(X,Y+16);
 X:=X+50;
 DRAWBLOCK(HATOM,2,0,0,5,5,X,Y+22,MODE);
 WSTAT(X+2,Y+22+6,'+');
 DRAWBLOCK(HATOM,2,0,0,5,5,X,Y-22,MODE);
 WSTAT(X+2,Y-22+6,'+');
 GETSPACE;
END; (* NETHYDROGEN *)

(*-----------------------------------------------------------------------*)
PROCEDURE NETCARBONATE;
(*-----------------------------------------------------------------------*)
VAR X,Y : INTEGER;
BEGIN
  X:=1; Y:=110;
  INITTURTLE;
  WSTAT(XMIN,YMAX-10,'Net reaction of carbonate ion:-');
  DRAWBLOCK(CARBATE,4,0,0,32,32,X,Y,MODE); X:=X+50;
  DRAWPLUS(X,Y+16); X:=X+30;
  DRAWBLOCK(HATOM,2,0,0,5,5,X,Y+26,MODE);
  WSTAT(X+2,Y+26+6,'+');
  DRAWBLOCK(HATOM,2,0,0,5,5,X,Y,MODE);
  WSTAT(X+2,Y+6,'+'); X:=X+40;
  DRAWARROW(X,Y+16,30); X:=X+55;
  DRAWBLOCK(CARBATE,4,0,0,32,32,X,Y,MODE);
  DRAWBLOCK(HATOM,2,0,0,5,5,X+2,Y+30,MODE); (*show H *)
  DRAWBLOCK(HATOM,2,0,0,5,5,X+26,Y+44,MODE);(*show H *)
  DRAWLINE(X+3,Y+23,X+3,Y+27,WHITE2); (*add bond to H *)
  DRAWLINE(X+27,Y+35,X+27,Y+39,WHITE2);(*add bond to H *)
  DRAWLINE(X+27,Y+15,X+32,Y+15,BLACK2);
  DRAWLINE(X+27,Y+18,X+32,Y+18,BLACK2); (*remove neg. charges*)
  DOWNARROW(X+16,Y-10,30); Y:=Y-60;
  DRAWBLOCK(CO2,4,0,0,32,8,X-25,Y,MODE);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X+25,Y,MODE);
  GETSPACE;
END; (* NETCARBONATE *)

(*-----------------------------------------------------------------------*)
PROCEDURE NETREACTION;
(*-----------------------------------------------------------------------*)
VAR X1,X2,X3,X4,X5,X6,X7,MIDY,UP,DOWN : INTEGER;
   S1,S2:STRING;
```

```
(*..........................................................................*)
PROCEDURE SHOWTEXT;
(*..........................................................................*)
BEGIN
 FILLAREA(XMIN,XMAX,YMAX-25,YMAX,BLACK1);
 WSTAT(XMIN,YMAX-10,S1);
 WSTAT(XMIN,YMAX-22,S2);
END; (* SHOWTEXT *)

(*..........................................................................*)
PROCEDURE DISPLAY;
(*..........................................................................*)
BEGIN
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X3,MIDY-DOWN,MODE);
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X3,MIDY+UP,MODE);
                        (* erase H30+*)
  DRAWBLOCK(HATOM,2,0,0,5,5,X3,MIDY+8,MODE);
  WSTAT(X3+2,MIDY+8+6,'+');
  DRAWBLOCK(HATOM,2,0,0,5,5,X3,MIDY-16,MODE);
  WSTAT(X3+2,MIDY-16+6,'+');        (*display H+ *)
  DRAWPLUS(X6,MIDY);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X7,MIDY-DOWN,MODE);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X7,MIDY+UP+12,MODE);
 END;                                          (* erase H20  *)

 (*..........................................................................*)
 PROCEDURE SIMPLIFY;
 (*..........................................................................*)
 VAR S3:STRING; CH:CHAR;
 BEGIN
    S1:='Press <SPACE BAR> to simplify reaction- ';
    S2:='replacing hydronium with hydrogen ions.';
    WSTAT(0,18,S1);
    WSTAT(0,0,S2);
    GETSPACE;
    DISPLAY;
    FILLAREA(XMIN,XMAX,YMIN,YMIN+30,BLACK1);
    S1:='Press <SPACE BAR>  to continue.';
    S2:='Press  <H>  to replace hydrogen ions';
    S3:='           with hydronium ions.';
    WSTAT(0,23,S1);
    WSTAT(0,9,S2);
    WSTAT(0,0,S3);
    GETKEY(CH,[SPACE,'Q','H']);
    FILLAREA(XMIN,XMAX,YMIN,YMIN+30,BLACK1);
    CASE CH OF
      'H': BEGIN  DISPLAY;  SIMPLIFY;  END;
      'Q': EXIT(CONCLUSION);
      END; (*CASE*)
 END; (* SIMPLIFY *)

 (*..........................................................................*)
 PROCEDURE SHOWSPECTATOR;
 (*..........................................................................*)
    PROCEDURE DRAWCHLORIDE(CLX,CLY:INTEGER);
    BEGIN
```

```
      DRAWBLOCK(ATOM,4,0,0,32,32,CLX,CLY,MODE);
      WSTAT(CLX+18,CLY+22,'-');
      WSTAT(CLX+8,CLY+12,'Cl');
     END; (* DRAWCHLORIDE *)

  BEGIN
    S1:='If hydrochloric acid is used, then';
    S2:='chloride is a spectator ion.';
    SHOWTEXT;
    S1:='Press <SPACE BAR> to display ';
    S2:='spectator ion.';
    WSTAT(0,12,S1); (*display *)
    WSTAT(126,1,S2);
    GETSPACE;
    DRAWCHLORIDE(X3+10,MIDY+10);
    DRAWCHLORIDE(X3+10,MIDY-42);
    DRAWCHLORIDE(X7-8,MIDY+28);
    DRAWCHLORIDE(X7-8,MIDY-55);
    FILLAREA(XMIN,XMAX,YMIN,YMIN+30,BLACK1);
  END; (* SHOWSPECTATOR *)

BEGIN (* NETREACTION *)
  MIDY:=90;
  X1:=XMIN+1; (*position of CO2*)
  X2:=X1+45; (* position of plus *)
  X3:=X2+20; (* position of acid *)
  X4:=X3+40; (* position of arrow*)
  X5:=X4+55; (* position of CO2 & H2O *)
  X6:=X5+50; (* position of plus *)
  X7:=X6+35; (* position of H2O  *)
  UP:=6;
  DOWN:=32+6;
  INITION(MET,CATION);
  INITTURTLE;
  S1:='Reaction between a carbonate';
  S2:='and an acid';
  SHOWTEXT;
  DRAWBLOCK(CARBATE,4,0,0,32,32,X1,MIDY-16,MODE);
  DRAWBLOCK(CATION,2,0,0,16,16,X1,MIDY+16,MODE);
  DRAWPLUS(X2,MIDY);
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X3,MIDY-DOWN,MODE);
  DRAWBLOCK(ACIDMOL[1].SHAPE,4,0,0,32,32,X3,MIDY+UP,MODE);
  DRAWARROW(X4,MIDY,30);
  DRAWBLOCK(CO2,4,0,0,32,8,X5,MIDY+UP,MODE);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X5,MIDY-DOWN,MODE);
  DRAWPLUS(X6,MIDY);
  DRAWBLOCK(CATION,2,0,0,16,16,X7,MIDY-8,MODE);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X7,MIDY-DOWN,MODE);
  DRAWBLOCK(WATERMOL[1].SHAPE,4,0,0,32,24,X7,MIDY+UP+12,MODE);
  SIMPLIFY;
  SHOWSPECTATOR;
  GETSPACE;
END; (* NETREACTION *)
```

```
    BEGIN (* CONCLUSION *)
     NETHYDROGEN;
     NETCARBONATE;
     NETREACTION;
    END; (* CONCLUSION *)


BEGIN (* MICRO *)
 INITMICROSHAPES;
 EXPLAINSHAPES;
    REPEAT
       REACTION;
       IF NOT QUIT THEN CONCLUSION;
       IF NOT QUIT THEN SOLVATECATIONS;
    UNTIL FIN('MICRO');
END;


(* CARB4 to be included in CARBONATE *)
(*************************************************************)
PROCEDURE MACRO;
(*************************************************************)
CONST TTUBEX=120;                  (* coord of test tube   *)
      TTUBEY=36;
      TTWIDTH=32;
      TTSIZE=116;                  (* ht of test tube      *)
      TTLEVEL=113;                 (* level of soln in tube*)
      SALTX=128;                   (* x coord of salt      *)
TYPE
  BITSHAPE=PACKED ARRAY[0..5,0..15]OF BOOLEAN;
  BUBSHAPE=PACKED ARRAY[0..5,0..7]OF BOOLEAN;
  FLSHAPE=PACKED ARRAY[0..7,0..7]OF BOOLEAN;
VAR
  SPACEPR : BOOLEAN;                 (* flag indicating space bar      *)
  SALTNUM:INTEGER;                   (* index to current shape of salt *)
  SALT:ARRAY [1..3] OF BITSHAPE;     (* various shapes of salt *)
  SALTY :INTEGER;                    (* y coord of salt        *)
  BUBBLE:BUBSHAPE;                   (*  shape of bubbles      *)
  FLAME:FLSHAPE;                     (* shape of flame on match*)
  BUBLX,BUBLY,                       (* coord of top bubble in test tube *)
  TESTX,TESTY : INTEGER;             (*coord of top buble in limewater*)

    (*=========================================================*)
    PROCEDURE INITVARS;
    (*=========================================================*)
      (*-------------------------------------------------------*)
      PROCEDURE INITSALT;
      (*-------------------------------------------------------*)
      (* SET UP SHAPES  OF CARBONATE*)
        (*.....................................................*)
        PROCEDURE INITBITS(ROW:INTEGER;VAR BITS:BITSHAPE;   S:STRING);
        (*.....................................................*)
        VAR   COL:INTEGER;
        BEGIN
          FOR COL:=0 TO 15 DO  BITS[ROW,COL]:=S[COL+1]='X';
        END; (* INITBIT *)
```

```
BEGIN (*INITSALT*)
   INITBITS(5,SALT[1],' XXX    XXXXXX  ');
   INITBITS(4,SALT[1],' X XX XXX    XXX');
   INITBITS(3,SALT[1],'X    X         X');
   INITBITS(2,SALT[1],'X              X');
   INITBITS(1,SALT[1],'X        XXXX  X');
   INITBITS(0,SALT[1],' XXXXXXXX  XXXX ');

   INITBITS(5,SALT[2],'                ');
   INITBITS(4,SALT[2],'  XX     XXX    ');
   INITBITS(3,SALT[2],' X  X  XX   X   ');
   INITBITS(2,SALT[2],'  X  XX     X   ');
   INITBITS(1,SALT[2],'   X   XXXX     ');
   INITBITS(0,SALT[2],'   XXXX         ');

   INITBITS(5,SALT[3],'                ');
   INITBITS(4,SALT[3],'                ');
   INITBITS(3,SALT[3],'   X    X       ');
   INITBITS(2,SALT[3],'  X X XX X      ');
   INITBITS(1,SALT[3],'   X X XX       ');
   INITBITS(0,SALT[3],'    XXX         ');
END; (* INITSALT *)

(*-----------------------------------------------------------------*)
PROCEDURE INITBUBBLE;
(*-----------------------------------------------------------------*)
  (*.............................................................*)
  PROCEDURE INIT(ROW:INTEGER;VAR BITS:BUBSHAPE;S:STRING);
  (*.............................................................*)
  VAR COL:INTEGER;
  BEGIN
    FOR COL:=0 TO 7 DO BITS[ROW,COL]:=S[COL+1]='X';
  END; (* INITBIT *)

BEGIN (*INITBUBBLE*)
  INIT(5,BUBBLE,'  XX   ');
  INIT(4,BUBBLE,' X  X  ');
  INIT(3,BUBBLE,'X    X ');
  INIT(2,BUBBLE,'X    X ');
  INIT(1,BUBBLE,' X  X  ');
  INIT(0,BUBBLE,'  XX   ');
END; (* INITBUBBLE *)

(*-----------------------------------------------------------------*)
PROCEDURE INITFLAME;
(*-----------------------------------------------------------------*)
  (*.............................................................*)
  PROCEDURE INITFL(ROW:INTEGER;VAR BITS:FLSHAPE;S:STRING);
  (*.............................................................*)
  VAR COL:INTEGER;
  BEGIN
   FOR COL:=0 TO 7 DO BITS[ROW,COL]:=S[COL+1]='X';
  END; (* INITBIT *)
```

```
 BEGIN (*INITFLAME*)
    INITFL(7,FLAME,'  XX  ');
    INITFL(6,FLAME,'  XX  ');
    INITFL(5,FLAME,' X X  ');
    INITFL(4,FLAME,' X X  ');
    INITFL(3,FLAME,'X   X ');
    INITFL(2,FLAME,'X   X ');
    INITFL(1,FLAME,' X X  ');
    INITFL(0,FLAME,'  XX  ');
 END; (* INITFLAME *)

BEGIN
  INITSALT;
  INITBUBBLE;
  INITFLAME;
END; (* INITVARS *)

(*==========================================================*)
PROCEDURE DRAWTTUBE(TUBEX,TUBEY,WIDTH,SIZE,LEVEL:INTEGER;
                                        COL: SCREENCOLOR);
(*==========================================================*)
VAR  EIGHTH,SIXNTH,RWIDTH:REAL;
     X,Y:ARRAY[1..15]OF INTEGER;  J: INTEGER;
BEGIN
 RWIDTH:=WIDTH;
 EIGHTH:=RWIDTH/8;
 Y[8]:=ROUND(EIGHTH*1.5);
 Y[4]:=ROUND(EIGHTH*1.25);
 Y[12]:=Y[4];
 Y[2]:=ROUND(EIGHTH*0.75);
 Y[14]:=Y[2];
 Y[1]:=ROUND(EIGHTH*0.5);
 Y[15]:=Y[1];
 Y[3]:=(Y[2]+Y[4])DIV 2;
 Y[13]:=Y[3];
 Y[6]:=ROUND(EIGHTH*1.4);
 Y[10]:=Y[6];
 Y[5]:=ROUND(EIGHTH*1.32);
 Y[11]:=Y[5];
 Y[7]:=Y[8];
 Y[9]:=Y[8];

 SIXNTH:=RWIDTH/16;
 FOR J:=1 TO 15 DO  X[J]:=ROUND(SIXNTH*J);
 MOVECOL(TUBEX,TUBEY+SIZE,COL);
 MOVETO(TUBEX,TUBEY);
 FOR J:=1 TO 15 DO  MOVETO(TUBEX+X[J],TUBEY-Y[J]);
 MOVETO(TUBEX+WIDTH,TUBEY);
 MOVETO(TUBEX+WIDTH,TUBEY+SIZE);
 MOVETO(TUBEX+WIDTH,LEVEL);
 MOVECOL(TUBEX,LEVEL,NONE);
END; (* DRAWTTUBE *)
```

```
(*=================================================================*)
PROCEDURE DROPSALT(VAR Y:INTEGER;BOTTOM:INTEGER);
(*=================================================================*)
VAR CH:CHAR;

  (*---------------------------------------------------------------*)
  PROCEDURE REQUEST;
  (*---------------------------------------------------------------*)
  VAR S:STRING;
  BEGIN
    WSTAT(160,Y,'Calcium carbonate');
    WSTAT(30,5,'Press <SPACE BAR> to add carbonate');
  END; (* REQUEST *)

BEGIN (*DROPSALT*)
  DRAWBLOCK(SALT[1],2,0,0,16,6,SALTX,Y,MODE); (*display metal*)
  REQUEST;
  GETKEY(CH,[SPACE,'Q']);
  REQUEST;
  REPEAT
    DRAWBLOCK(SALT[1],2,0,0,16,6,SALTX,Y,MODE);(* erase *)
    Y:=Y-10;     (* CALC. NEW HEIGHT*)
    DRAWBLOCK(SALT[1],2,0,0,16,6,SALTX,Y,MODE);(* display *)
    DELAY(15);
  UNTIL Y<=(BOTTOM+4);
END; (* DROPSALT *)



(*CARB5*)
 (*================================================================*)
 PROCEDURE REACTION;
 (*================================================================*)
 CONST PRESS='Press <SPACE BAR> ';
 VAR
   TTUBEX2,          (* x coor of tube with limewater*)
   LOWLEVEL,         (* min y coord for drawing bubbles*)
   TESTLEV,          (* min y coord for bubbles in limewater *)
   MATCHX,MATCHY,            (* coord of match      *)
   DY,             (* no. pixels bubbles rise each cycle*)
   GAP,            (* spacing between each bubble      *)
   SPEED: INTEGER; (* wait period between moving bubbles*)
   CH:CHAR;

   (*------------------------------------------------------------*)
   PROCEDURE INIT;
   (*------------------------------------------------------------*)
   CONST SKIP=4; (*determines spacing between bubbles *)
   BEGIN
     MATCHX:=62;MATCHY:=182;   (*starting coord of match*)
     BUBLX:=132;BUBLY:=TTUBEY+2;(*starting coord of bubble*)
     LOWLEVEL:=TTUBEY+2;
     DY:=4; GAP:=SKIP*DY; (* required for MOVEBUBBLES *)
     SALTNUM:=1;  (*index to shape of salt *)
   END; (* INIT *)
```

```
(*----------------------------------------------------------------------*)
PROCEDURE CHECKKEY(VAR SP:BOOLEAN);
(*----------------------------------------------------------------------*)
VAR CH:CHAR;
BEGIN
  READ(CH);
  QUIT:=((CH='Q') OR (CH='q'));
  SP:=CH=SPACE;
  IF QUIT THEN EXIT(REACTION);
END; (* CHECKKEY *)

(*----------------------------------------------------------------------*)
PROCEDURE DRAWMATCH(X,Y:INTEGER;COL:SCREENCOLOR);
(*----------------------------------------------------------------------*)
BEGIN
 MOVECOL(X-12,Y+8,COL);
 MOVETO(X,Y);
 MOVECOL(X,Y+4,NONE);
 DRAWBLOCK(FLAME,2,0,0,8,8,X-3,Y+6,MODE);
END; (* DRAWMATCH *)

(*----------------------------------------------------------------------*)
PROCEDURE ERASEMATCH(X,Y:INTEGER);
(*----------------------------------------------------------------------*)
BEGIN
 MOVECOL(X-12,Y+8,BLACK2);
 MOVETO(X,Y);
 MOVECOL(X,Y+4,NONE);
END; (* ERASEMATCH *)

(*----------------------------------------------------------------------*)
PROCEDURE MOVEMATCH(VAR MATCHX,MATCHY,NUM:INTEGER);
(*----------------------------------------------------------------------*)
BEGIN
 DRAWMATCH(MATCHX,MATCHY,BLACK2);
 MATCHX:=MATCHX+8;
 MATCHY:=MATCHY-3;
 DRAWMATCH(MATCHX,MATCHY,WHITE2);
 NUM:=0;
END; (* MOVEMATCH *)

(*----------------------------------------------------------------------*)
PROCEDURE STATEMENT;
(*----------------------------------------------------------------------*)
VAR S1,S2:STRING;
BEGIN
 S1:='A carbonate dissolves in HCl solution';
 S2:='and a gas is evolved';
 WSTAT(20,13,S1);
 WSTAT(30,1,S2);
END; (* STATEMENT *)

(*----------------------------------------------------------------------*)
PROCEDURE RESULT;
(*----------------------------------------------------------------------*)
VAR S1,S2:STRING;
```

```
BEGIN
 S1 :='A non combustible gas is';
 S2 :='produced when the carbonate dissolves';
 WSTAT(30,13,S1);
 WSTAT(10,1,S2);
END; (* RESULT *)

(*-----------------------------------------------------------------*)
PROCEDURE DRAWBUBBLES(X,Y:INTEGER);
(*-----------------------------------------------------------------*)
   (*..............................................................*)
   PROCEDURE EXTRABUBBLES(X,NEWY:INTEGER);
   (*..............................................................*)
   VAR MORE:BOOLEAN;
   BEGIN
    REPEAT
      NEWY:=NEWY-GAP;
      MORE:=NEWY>=LOWLEVEL;
      IF MORE THEN  DRAWBLOCK(BUBBLE,2,0,0,8,6,X,NEWY,MODE)
    UNTIL NOT MORE;
   END; (* EXTRABUBBLES *)

BEGIN (* DRAWBUBBLES *)
   DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE);
   EXTRABUBBLES(X,Y);
END; (* DRAWBUBBLES *)

(*-----------------------------------------------------------------*)
PROCEDURE MOVEBUBBLES(VAR X,CURRENTY:INTEGER);
(*-----------------------------------------------------------------*)
VAR Y:INTEGER;
BEGIN
 Y:=CURRENTY;
 CURRENTY:=CURRENTY+DY;
 IF CURRENTY>=TTLEVEL THEN CURRENTY:=CURRENTY-GAP;
 REPEAT
    DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y,MODE);  (* erase *)
    IF (Y+DY)<TTLEVEL THEN DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE);
    Y:=Y-GAP;                                          (*display*)
 UNTIL Y<LOWLEVEL;
 IF (Y+DY)>=LOWLEVEL THEN   DRAWBLOCK(BUBBLE,2,0,0,8,6,X,Y+DY,MODE);
END; (* MOVEBUBBLES *)                  (*display new bubble at bottom*)

(*-----------------------------------------------------------------*)
PROCEDURE BUBCYCLE(CYCLENUM:INTEGER);
(*-----------------------------------------------------------------*)
VAR CYCLES:INTEGER;
BEGIN
 CYCLES:=0;
 REPEAT
   CYCLES:=CYCLES+1;
   MOVEBUBBLES(BUBLX,BUBLY);
   DELAY(SPEED);
   IF KEYIN THEN CHECKKEY(SPACEPR);
 UNTIL ((CYCLES>=CYCLENUM) OR SPACEPR);
END; (* BUBCYCLE *)
```

```
(*-----------------------------------------------------------------*)
PROCEDURE PAINT(X,Y,TOP:INTEGER; COL:SCREENCOLOR);
(*-----------------------------------------------------------------*)
  (*...............................................................*)
  FUNCTION LOWEST(VAR Y:INTEGER):INTEGER;
  (*...............................................................*)
  BEGIN
   WHILE (NOT SCREENBIT(X,Y)) AND  (NOT SCREENBIT(X+1,Y)) AND (Y>0)
     DO  Y:=Y-1;
   Y:=Y+1;
   LOWEST:=Y;
  END; (* LOWEST *)

  (*...............................................................*)
  FUNCTION RIGHTMOST(X:INTEGER):INTEGER;
  (*...............................................................*)
  BEGIN
   WHILE (NOT SCREENBIT(X,Y)) AND (X<XMAX) DO X:=X+1;
   RIGHTMOST:=X-1;
  END; (* RIGHTMOST *)

  (*...............................................................*)
  FUNCTION LEFTMOST(X: INTEGER):INTEGER;
  (*...............................................................*)
  BEGIN
     WHILE (NOT SCREENBIT(X,Y)) AND (X>0) DO X:=X-1;
     LEFTMOST:=X+1;
  END; (* LEFTMOST *)

BEGIN (* PAINT *)
 MOVETO(X,LOWEST(Y));
 WHILE (NOT SCREENBIT(X,Y)) AND (NOT SCREENBIT(X+1,Y))  AND (Y<TOP) DO
   BEGIN
     MOVETO(X,Y);
     MOVETO(RIGHTMOST(X),Y);
     PENCOLOR(COL);
     MOVETO(LEFTMOST(X),Y);
     PENCOLOR(NONE);   Y:=Y+1;
   END;
END; (* PAINT *)

(*-----------------------------------------------------------------*)
PROCEDURE MILKY;
(*-----------------------------------------------------------------*)
  (*...............................................................*)
  PROCEDURE FILL(X1,X2,TOP,BOTTOM:INTEGER;   COL:SCREENCOLOR);
  (*...............................................................*)
  BEGIN
   MOVECOL(X1,BOTTOM,COL);
   REPEAT
      MOVETO(X1,BOTTOM);
      MOVETO(X2,BOTTOM);
      BOTTOM:=BOTTOM+1;
   UNTIL BOTTOM=TOP;
   PENCOLOR(NONE);
  END; (* FILL *)
```

```
BEGIN (* MILKY *)
  PAINT(TTUBEX2+(TTWIDTH DIV 2),TTUBEY-2,TTUBEY,WHITE);
  FILL(TTUBEX2+1,TTUBEX2+TTWIDTH-1,TTLEVEL,TTUBEY,WHITE);
END; (* MILKY *)

(*------------------------------------------------------------------*)
PROCEDURE SWAP(VAR CURRENT:INTEGER);
(*------------------------------------------------------------------*)
BEGIN
  DRAWBLOCK(SALT[CURRENT],2,0,0,16,6,SALTX,SALTY,MODE);
  CURRENT:=CURRENT+1;
  DRAWBLOCK(SALT[CURRENT],2,0,0,16,6,SALTX,SALTY,MODE);
END;(* SWAP *)

(*------------------------------------------------------------------*)
PROCEDURE CO2RESULT;
(*------------------------------------------------------------------*)
  (*................................................................*)
  PROCEDURE INFORMCO2;
  (*................................................................*)
  VAR S1,S2:STRING;
  BEGIN
    S1:='Lime water turned milky-';
    S2:='Gas must be carbon dioxide';
    WSTAT(30,13,S1);   WSTAT(30,1,S2);
  END; (* INFORMCO2 *)

BEGIN (* CO2RESULT *)
  MILKY;
  RESULT; (* erase *)
  INFORMCO2;
END; (* CO2RESULT *)

(*------------------------------------------------------------------*)
PROCEDURE TESTCO2;
(*------------------------------------------------------------------*)
VAR COUNT:INTEGER;
  (*................................................................*)
  PROCEDURE PROMPT;
  (*................................................................*)
  BEGIN
    WSTAT(0,183,CONCAT(PRESS,'for limewater test'));
  END; (* PROMPT *)

  (*................................................................*)
  PROCEDURE DRAWTUBING(X1,X2,Y,SIZE,WIDTH: INTEGER;ACOL:SCREENCOLOR);
  (*................................................................*)
  CONST TWIDTH=4;
  VAR  HEIGHT: INTEGER;
      PROCEDURE TUBE;
      BEGIN
        MOVECOL(X1,Y,ACOL);
        MOVETO(X1,Y+HEIGHT);
        MOVETO(X2,Y+HEIGHT);
        MOVECOL(X2,Y-60,NONE);
      END; (* TUBE *)
```

```
BEGIN (* DRAWTUBING *)
   Y:=Y+SIZE-20;
   X1:=X1+(WIDTH DIV 2);
   X2:=X2+(WIDTH DIV 2)+TWIDTH;
   HEIGHT:=34;
   TUBE;
   X1:=X1+TWIDTH;
   X2:=X2-TWIDTH;
   HEIGHT:=HEIGHT+TWIDTH;
   TUBE;
END; (* DRAWTUBING *)

(*.........................................................................................*)
PROCEDURE SHOWLIMEWATER;
(*.........................................................................................*)

   PROCEDURE DRAWSTOPPER(X,Y,WIDTH : INTEGER;  ACOL:SCREENCOLOR);
   CONST  HT=15;
   BEGIN
     MOVECOL(X,Y,ACOL);
     MOVETO(X,Y+HT);
     MOVETO(X+WIDTH,Y+HT);
     MOVETO(X+WIDTH,Y);
     MOVECOL(X,Y,NONE);
     FILLAREA(X+2,X+WIDTH-2,Y+1,Y+HT-1,BLACK1);
   END; (* DRAWSTOPPER *)

BEGIN (*SHOWLIMEWATER*)
  TTUBEX2:=TTUBEX-60;
  DRAWTTUBE(TTUBEX2,TTUBEY,TTWIDTH,TTSIZE,TTLEVEL,WHITE1);
  WSTAT(TTUBEX2-40,TTUBEY+40,'lime');
  WSTAT(TTUBEX2-40,TTUBEY+30,'water');
  DRAWTUBING(TTUBEX,TTUBEX-60,TTUBEY,TTSIZE,TTWIDTH,WHITE1);
  DRAWSTOPPER(TTUBEX+2,TTUBEY+TTSIZE-10,TTWIDTH-4,WHITE1);
  TESTX:=TTUBEX-60+(TTWIDTH DIV 4);
  TESTY:=TTUBEY+30;
  TESTLEV:=TESTY;
END; (* SHOWLIMEWATER *)

BEGIN (* TESTCO2 *)
 PROMPT; (*display prompt*)
 BUBCYCLE(10000);
 PROMPT; (* erase*)
 ERASEMATCH(MATCHX,MATCHY); (*erase match*)
 SHOWLIMEWATER;
 LOWLEVEL:=TESTLEV;
 DRAWBUBBLES(TESTX,TESTY);(* bubbles in limewater*)
 LOWLEVEL:=TTUBEY+2;
 COUNT:=0;
 REPEAT
   MOVEBUBBLES(BUBLX,BUBLY);
   IF KEYIN THEN CHECKKEY(SPACEPR);
   LOWLEVEL:=TESTLEV;
   MOVEBUBBLES(TESTX,TESTY);
   LOWLEVEL:=TTUBEY+2;
   COUNT:=COUNT+1;
```

```
    IF COUNT=25 THEN SWAP(SALTNUM);
   UNTIL ((COUNT>=50) OR QUIT);
   DRAWBLOCK(SALT[3],2,0,0,16,6,SALTX,SALTY,MODE); (* erase salt  *)
   DRAWBUBBLES(BUBLX,BUBLY);(*erase bubbles in carbonate*)
   LOWLEVEL :=TESTLEV;
   DRAWBUBBLES(TESTX,TESTY);(*erase bubbles in limewater*)
   CO2RESULT;
  END; (* TESTCO2 *)

 (*----------------------------------------------------------------------*)
 PROCEDURE FLAMETEST;
 (*----------------------------------------------------------------------*)
 VAR J: INTEGER;
   (*.....................................................................*)
   PROCEDURE PROMPT;
   (*.....................................................................*)
   BEGIN
      WSTAT(40,183,CONCAT(PRESS,'for flame test'));
   END; (* REQUEST *)

  BEGIN
   PROMPT; (* prompt test gas with flame *)
   BUBCYCLE(10000);
   PROMPT; (* erase prompt *)
   SPACEPR :=FALSE;
   DRAWMATCH(MATCHX,MATCHY,WHITE2);
   REPEAT
     MOVEBUBBLES(BUBLX,BUBLY);
     MOVEMATCH(MATCHX,MATCHY,J);
     DELAY(SPEED);
   UNTIL MATCHX>=BUBLX;
   DRAWBLOCK(FLAME,2,0,0,8,8,MATCHX-3,MATCHY+6,MODE);
  END; (* FLAMETEST *)              (*erase*)

BEGIN  (* REACTION *)
 INIT;
 SPACEPR :=FALSE;
 SPEED :=10;
 STATEMENT; (*display *)
 DRAWBUBBLES(BUBLX,BUBLY);  (* display *)
 FLAMETEST;
 STATEMENT; (* erase *)
 IF NOT QUIT THEN RESULT;
 BUBCYCLE(4);
 SWAP(SALTNUM);
 TESTCO2;
 GETKEY(CH,[SPACE,'Q']);
END; (* REACTION *)
```

```
(*CARB6*)
  (*=================================================================*)
  PROCEDURE CONCLUSION;
  (*=================================================================*)
  CONST WIDTH=8;

    (*---------------------------------------------------------------*)
    PROCEDURE BIGSTAT(Y:INTEGER; S:STRING);
    (*---------------------------------------------------------------*)
    VAR  LETTER,NUM,X :INTEGER;  CH:CHAR;
    BEGIN
     X:=(XMAX-(10*LENGTH(S))) DIV 2;
     FOR LETTER:=1 TO LENGTH(S) DO
      BEGIN
       MOVETO(X,Y);
       NUM:=ORD(S[LETTER]);
       IF NUM IN [65..90] THEN NUM:=NUM-65 (* A..Z *)
         ELSE IF NUM=43 THEN NUM:=26; (* + *)
       WCHAR(CHR(NUM));
       X:=X+10;
      END;
    END; (* BIGSTAT *)

    (*---------------------------------------------------------------*)
    PROCEDURE ENCLOSE(WIDTH:INTEGER; COL:SCREENCOLOR);
    (*---------------------------------------------------------------*)
    BEGIN
      FILLAREA(XMIN,XMIN+WIDTH,YMIN,YMAX,COL);
      FILLAREA(XMAX-WIDTH,XMAX,YMIN,YMAX,COL);
      FILLAREA(XMIN,XMAX,YMIN,YMIN+WIDTH,COL);
      FILLAREA(XMIN,XMAX,YMAX-WIDTH,YMAX,COL);
    END; (* ENCLOSE *)

    (*---------------------------------------------------------------*)
    PROCEDURE DOWNARROW(X,Y1,Y2:INTEGER; COL:SCREENCOLOR);
    (*---------------------------------------------------------------*)
    CONST WID=3;
    VAR X1,X2:INTEGER;
    BEGIN
     FILLAREA(X-WID,X+WID,Y2,Y1,COL);
     X1:=X-3*WID;
     X2:=X+3*WID;
     REPEAT
       DRAWLINE(X1,Y2,X2,Y2,COL);
       Y2:=Y2-1;
       X1:=X1+1;
       X2:=X2-1;
     UNTIL X1>=X2;
    END; (* DOWNARROW *)

    (*---------------------------------------------------------------*)
    PROCEDURE TEXT;
    (*---------------------------------------------------------------*)
    TYPE METAL=(NA,CA,MG);
         ANACID=(CHLORIC,SULF);
    VAR  BAND:INTEGER;    AMETAL:METAL; ANYACID:ANACID;
```

```
(*......................................................................................*)
PROCEDURE GENERAL;
(*......................................................................................*)
VAR Y:INTEGER; CH:CHAR;
BEGIN
 Y:=YMAX-35;
 BIGSTAT(Y,'ACID + CARBONATE');  Y:=Y-20;
 DOWNARROW(XMAX DIV 2,Y,Y-25,WHITE);  Y:=Y-60;
 BIGSTAT(Y,'CARBON DIOXIDE + WATER');   Y:=Y-25;
 BIGSTAT(Y,'+');    Y:=Y-25;
 BIGSTAT(Y,'SALT');
 GETKEY(CH,[SPACE,'Q']);
 IF QUIT THEN EXIT(TEXT);
END; (* GENERAL *)

(*......................................................................................*)
PROCEDURE EQUATION(ACID:ANACID);
(*......................................................................................*)
VAR Y:INTEGER; CH:CHAR;
    S1,S2,METSTR:STRING;
BEGIN
 METSTR:='SODIUM';
 CASE ACID OF
   CHLORIC: BEGIN S1:='HYDROCHLORIC ACID'; S2:=' CHLORIDE'; END;
   SULF   :  BEGIN S1:='SULFURIC ACID'; S2:=' SULFATE'; END;
   END; (*CASE*)
 Y:=YMAX-30;
 BIGSTAT(Y,S1); Y:=Y-15;
 BIGSTAT(Y,'+'); Y:=Y-15;
 BIGSTAT(Y,CONCAT(METSTR,' CARBONATE'));  Y:=Y-15;
 DOWNARROW(XMAX DIV 2,Y,Y-20,WHITE);   Y:=Y-55;
 BIGSTAT(Y,'CARBON DIOXIDE + WATER');   Y:=Y-20;
 BIGSTAT(Y,'+');   Y:=Y-20;
 BIGSTAT(Y,CONCAT(METSTR,S2));
 GETKEY(CH,[SPACE,'Q']);
 IF QUIT THEN EXIT(TEXT);
END; (* EQUATION *)

(*......................................................................................*)
PROCEDURE CHANGE(MET:METAL; ACID:ANACID);
(*......................................................................................*)
VAR Y:INTEGER; CH:CHAR;
   SALT:STRING[10];
   METSTR:STRING[10];

   PROCEDURE BLANKLINE(Y,X1,X2:INTEGER);
   CONST BLANK='          ';
   BEGIN
    WHILE X1<X2 DO
     BEGIN
       WSTAT(X1,Y,BLANK);
       X1:=X1+70
     END;
   END; (* BLANKLINE *)
```

```
      BEGIN (* CHANGE *)
       CASE ACID OF
        CHLORIC : SALT :=' CHLORIDE';
        SULF    : SALT :=' SULFATE';
        END; (*CASE*)
       CASE MET OF
        NA: METSTR :='SODIUM';
        CA: METSTR :='CALCIUM';
        MG: METSTR :='MAGNESIUM';
        END; (*CASE*)
       Y :=YMAX-60;
       BLANKLINE(Y,XMIN+20,XMAX-60);
       BIGSTAT(Y,METSTR);  Y :=Y-110;
       BLANKLINE(Y,XMIN+20,XMAX-60);
       BIGSTAT(Y,CONCAT(METSTR,SALT));
       GETKEY(CH,[SPACE,'Q']);
       IF QUIT THEN EXIT(TEXT);
      END; (* CHANGE *)

    BEGIN
     GENERAL;
     BAND :=WIDTH+5;
     FOR ANYACID :=CHLORIC TO SULF DO
      BEGIN
        FILLAREA(XMIN+BAND,XMAX-BAND,YMIN+BAND,YMAX-BAND,BLACK);
        EQUATION(ANYACID);
        FOR AMETAL :=CA TO MG DO CHANGE(AMETAL,ANYACID);
      END;
    END; (* TEXT *)

  BEGIN (* CONCLUSION *)
   CHARTYPE(10);
   INITTURTLE;
   ENCLOSE(WIDTH,VIOLET);
   TEXT;
   CHARTYPE(6)
  END; (* CONCLUSION *)


BEGIN (* MACRO *)
 INITVARS;
 REPEAT
   INITTURTLE;
   DRAWTTUBE(TTUBEX,TTUBEY,TTWIDTH,TTSIZE,TTLEVEL,WHITE1);
   WSTAT(TTUBEX+TTWIDTH+10,TTUBEY+40,'dilute');
   WSTAT(TTUBEX+TTWIDTH+18,TTUBEY+30,'HCl');
   SALTY :=182;
   DROPSALT(SALTY,TTUBEY);
   IF NOT QUIT THEN REACTION;
   IF NOT QUIT THEN CONCLUSION;
 UNTIL FIN('MACRO');
 TEXTMODE;
END;  (* MACRO *)
```

```
(****************************************************************)
PROCEDURE SELECT(VAR CH:CHAR);
(****************************************************************)
CONST DEMO='SCOPIC demonstration    ......(';
VAR  X,Y: INTEGER;
BEGIN
   TEXTMODE;
   PAGE(OUTPUT);
   X:=0; Y:=1;
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
   WRITE(AT(X,Y),'REACTION BETWEEN A CARBONATE & AN ACID'); Y:=Y+2;
   WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+4;;
   WRITE(AT(X,Y),'MACRO',DEMO,'1)'); Y:=Y+3;
   WRITE(AT(X,Y),'MICRO',DEMO,'2)'); Y:=Y+3;
   WRITE(AT(X,Y),'QUIT - back to main menu     ......(Q)'); Y:=Y+4;
   WRITE(AT(X+10,Y),'Select option  ..........(  )');
   GETTEXTCHAR(37,Y,CH,['1','2','Q']);
   QUIT:=CH='Q';
   PAGE(OUTPUT);
 END; (* SELECT *)



 BEGIN  (* MAIN *)
   SETCHAIN(' :DEMOMENU');
   CHARTYPE(MODE);
   FALSEARRAY(BLANK);
   INITSHAPES; (* Init shapes for micro which takes a while*)
   REPEAT
     SELECT(OPTION);
     IF NOT QUIT THEN
      BEGIN
       CASE OPTION OF
         '1' : MACRO;
         '2' : MICRO;
         END; (*CASE*)
       QUIT:=FALSE;
      END;
   UNTIL QUIT;
 END. (*CARBONATE*)
```

```
(*$S++*)
PROGRAM LITMUS;
USES CHAINSTUFF,TURTLEGRAPHICS,USEFUL;

CONST MODE=6;
VAR
  QUIT,COLOUR  : BOOLEAN;
  OPTION : CHAR;

(**********************************************************)
PROCEDURE GETKEY(VAR ACH:CHAR; LEGALSET:CHARSET);
(**********************************************************)
BEGIN
  GETACHAR(ACH,LEGALSET);
  QUIT:=(ACH='Q');
END; (* GETKEY *)

(**********************************************************)
PROCEDURE MICRO;
(**********************************************************)
TYPE
  BIGSIZE=PACKED ARRAY[1..32,1..32] OF BOOLEAN;
  MEDSIZE=PACKED ARRAY[1..16,1..16] OF BOOLEAN;
  SMSIZE=PACKED ARRAY[1..8,1..8]OF BOOLEAN;
  TINYSIZE=PACKED ARRAY[1..5,1..8] OF BOOLEAN;
  LITCOL=(BLUELIT,REDLIT);
  LITTYPE=RECORD
            X,Y,SIZE  : INTEGER;
            COL : SCREENCOLOR;
           END;
  PH=(NEUTRAL,ACIDIC,BASIC);

VAR
   SMALLH:TINYSIZE;                          (*shape of H atom *)
   BLANK,
   HYDROX,                                   (*shape of OH-    *)
   LARGACID,WATER: BIGSIZE;          (* shape of acid and water   *)
   BLUECOL, REDCOL : SCREENCOLOR;

    (*=======================================================*)
    PROCEDURE SMALLBITS(ROW:INTEGER; VAR BITS:SMSIZE; S:STRING);
    (*=======================================================*)
    VAR   COL:INTEGER;
    BEGIN
       FOR COL:=1 TO 8 DO  BITS[ROW,COL]:=S[COL]='X';
    END; (* SMALLBITS *)

    (*=======================================================*)
    PROCEDURE DRAWBOND(X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);
    (*=======================================================*)
    BEGIN
       MOVECOL(X1,Y1,COL);
       MOVECOL(X2,Y2,NONE);
    END; (* DRAWBOND *)
```

```
(*==================================================================*)
PROCEDURE ADDH(X,Y,SIZE:INTEGER; COL:SCREENCOLOR);
(*==================================================================*)
BEGIN
 X:=X+(3*SIZE)+2;  (*new x,y correspond to middle R.H side of shape*)
 Y:=Y+SIZE;        (*so that H atom can be attached at this pt*)
 MOVECOL(X,Y,COL);
 MOVECOL(X+(SIZE DIV 2),Y,NONE);
 WSTAT(X+(SIZE DIV 2)+2,Y-4,'H');
END; (* ADDH *)

(*==================================================================*)
PROCEDURE DRAWLITMUS(X,Y,SIZE:INTEGER; COL:SCREENCOLOR);
(*==================================================================*)
(*Display a large structure to represent a litmus molecule *)
VAR  START,STOP, J : INTEGER;
BEGIN
   START:=X;  STOP:=X+2*SIZE;  Y:=Y+2*SIZE;
   MOVETO(START,Y);
   FOR J:=1 TO SIZE DO
     BEGIN
       MOVECOL(START,Y,COL);
       MOVECOL(STOP,Y,NONE);
       START:=START-1;
       STOP:=STOP+1;
       Y:=Y-1;
     END;
  FOR J:=1 TO SIZE DO
    BEGIN
       MOVECOL(START,Y,COL);
       MOVECOL(STOP,Y,NONE);
       START:=START+1;
       STOP:=STOP-1;
       Y:=Y-1;
    END;
 END;  (* DRAWLITMUS *)

(*==================================================================*)
PROCEDURE DRAWEL(X,Y,SIZE:INTEGER; COL:SCREENCOLOR);
(*==================================================================*)
VAR YY,J: INTEGER;

   (*--------------------------------------------------------------*)
   PROCEDURE ADDPLUS(X,Y:INTEGER);
   (*--------------------------------------------------------------*)
   VAR  LEN,MIDLEN,J :INTEGER;
   BEGIN
    LEN:=(SIZE DIV 3) ;
    FOR J:=1  TO 2 DO
      BEGIN
        DRAWLINE(X,Y,X+LEN,Y,COL);
        Y:=Y+1;
      END;
    MIDLEN:=LEN DIV 2;
    DRAWLINE(X+MIDLEN,Y+MIDLEN,X+MIDLEN,Y-LEN ,COL);
   END; (* ADDPLUS *)
```

```
BEGIN (*DRAWEL*)
 X:=X+(SIZE DIV 2);
 YY:=Y+(SIZE DIV 2);
 IF COL=REDCOL THEN
    BEGIN
      COL:=BLACK1;
      ADDPLUS(X+SIZE,YY+SIZE);
    END  ELSE COL:=BLACK2;
 FOR J:=1 TO 3 DO
   BEGIN
    DRAWLINE(X,YY,X+SIZE,YY,COL); YY:=YY+1;
   END;
 FOR J:=1 TO 2 DO
  BEGIN
   DRAWLINE(X,YY,X,YY+SIZE,COL); X:=X+1;
  END;
END; (* DRAWEL *)

(*=============================================================*)
PROCEDURE INITSMH(VAR ANYARRAY:TINYSIZE);
(*=============================================================*)

  (*-----------------------------------------------------------*)
  PROCEDURE SMBITS(ROW:INTEGER; VAR BITS:TINYSIZE; S:STRING);
  (*-----------------------------------------------------------*)
  VAR  COL: INTEGER;
  BEGIN
   FOR COL:=1 TO 8 DO  BITS[ROW,COL]:=S[COL]='X';
  END; (* SMBITS *)

 BEGIN (* INITSMH *)
   SMBITS(5,ANYARRAY,'X    X    ');
   SMBITS(4,ANYARRAY,'X    X    ');
   SMBITS(3,ANYARRAY,'XXXXX     ');
   SMBITS(2,ANYARRAY,'X    X    ');
   SMBITS(1,ANYARRAY,'X    X    ');
 END; (* INITSMH *)

(*=============================================================*)
PROCEDURE NAME(X,Y,SIZE:INTEGER; ANYCOL:SCREENCOLOR);
(*=============================================================*)
(* labels type of litmus to right of the litmus molecule*)
VAR S:STRING;
BEGIN
  IF ANYCOL=REDCOL THEN S:=' RED' ELSE S:='BLUE';
  WSTAT(X+5*SIZE,Y+SIZE-4,CONCAT(S,' LITMUS'));
END;  (* NAME *)

(*=============================================================*)
PROCEDURE DISPLAYLITMUS;
(*=============================================================*)
VAR CH:CHAR;
     LITSHAPE:  ARRAY[BLUELIT..REDLIT] OF LITTYPE;
     ANYCOL:LITCOL;
```

```
  (*----------------------------------------------------------------*)
  PROCEDURE INITVARS;
  (*----------------------------------------------------------------*)
  BEGIN
    WITH LITSHAPE[REDLIT] DO
      BEGIN X:=60; Y:=60; SIZE:=16;COL:=REDCOL;  END;
    WITH LITSHAPE[BLUELIT] DO
      BEGIN  X:=60; Y:=120; SIZE:=16; COL:=BLUECOL; END;
   END; (* INITVARS *)

  (*----------------------------------------------------------------*)
  PROCEDURE EXPLAIN;
  (*----------------------------------------------------------------*)
  VAR  S1,S2 : STRING;
  BEGIN
    S1:='The two different colors result from';
    S2:='two slightly different structures';
    WSTAT(0,25,S1);
    WSTAT(0,16,S2);
  END; (* EXPLAIN *)


BEGIN (* DISPLAYLITMUS *)
 INITVARS;
 INITTURTLE;
 WSTAT(20,175,'Litmus exists in two forms:-');
 FOR ANYCOL:=BLUELIT TO REDLIT DO
   BEGIN
    WITH LITSHAPE[ANYCOL] DO
     BEGIN
       DRAWLITMUS(X,Y,SIZE,COL);
       DRAWEL(X,Y,SIZE,COL);
       IF ANYCOL=REDLIT THEN ADDH(X,Y,SIZE,WHITE1);
       NAME(X,Y,SIZE,COL);
     END;
   END;
 EXPLAIN;
 GETKEY(CH,[SPACE,'Q']);
END; (* DISPLAYLITMUS *)

(*================================================================*)
PROCEDURE CHANGELIT(X,Y,SIZE:INTEGER);
(*================================================================*)
VAR CH:CHAR;

  (*----------------------------------------------------------------*)
  PROCEDURE INFORM(ANYCOL:SCREENCOLOR);
  (*----------------------------------------------------------------*)
  VAR  S1,S2,S3: STRING;
  BEGIN
   IF ANYCOL=REDCOL THEN
    BEGIN
      S1:='If the hydrogen ion is removed from ';
      S2:='RED litmus then its colour';
      S3:='changes to BLUE';
    END
```

```
      ELSE
       BEGIN
          S1 :='If a hydrogen ion is added to BLUE';
          S2:='litmus then its colour changes ';
          S3:='to RED';
       END;
    WSTAT(0,17,S1); WSTAT(0,8,S2); WSTAT(0,0,S3);
   END; (* INFORM *)

   (*-----------------------------------------------------------------*)
   PROCEDURE CHANGECOL(OLDCOL:SCREENCOLOR);
   (*-----------------------------------------------------------------*)
   VAR NEWCOL:SCREENCOLOR;
   BEGIN
     IF OLDCOL=REDCOL THEN NEWCOL:=BLUECOL ELSE  NEWCOL:=REDCOL;
     DRAWLITMUS(X,Y,SIZE,NEWCOL);
     DRAWEL(X,Y,SIZE,NEWCOL);
     NAME(X,Y,SIZE,OLDCOL);  (* erase *)
     NAME(X,Y,SIZE,NEWCOL);
   END; (* CHANGECOL *)

 BEGIN   (* CHANGELIT *)
  INITTURTLE;
  DRAWLITMUS(X,Y,SIZE,BLUECOL);
  DRAWEL(X,Y,SIZE,BLUECOL);
  NAME(X,Y,SIZE,BLUECOL);
  INFORM(BLUECOL);
  GETKEY(CH,[SPACE,'Q']);
  IF NOT QUIT THEN
   BEGIN
     ADDH(X,Y,SIZE,WHITE1);
     CHANGECOL(BLUECOL);
     GETKEY(CH,[SPACE,'Q']);
     IF NOT QUIT THEN
      BEGIN
        INFORM(BLUECOL);  (*erase*)
        INFORM(REDCOL);   (*display*)
        GETKEY(CH,[SPACE,'Q']);
        IF NOT QUIT THEN
          BEGIN
            ADDH(X,Y,SIZE,BLACK2);
            CHANGECOL(REDCOL);
            GETKEY(CH,[SPACE,'Q']);
          END;
      END;
   END;
 END; (* CHANGELIT *)

 (*==============================================================*)
 PROCEDURE DEFINEMOLEC(VAR NEWARRAY:BIGSIZE; ACIDITY:PH; SHAPE:INTEGER);
 (*==============================================================*)
 (*ACIDITY DETERMINES WHICH MOLECULE IS TO BE DISPLAYED,
   SHAPE IS REQUIRED TO DISTINGUISH BETWEEN 2 ORIENTATIONS OF WATER *)
 VAR
   H1X,H1Y,H2X,H2Y,H3X,H3Y,OX,OY:INTEGER;
   MAXCOL,HEIGHT,WIDTH: INTEGER;
```

```
(*-----------------------------------------------------------------*)
PROCEDURE INIT(WIDTH,HEIGHT :INTEGER;SYMBOL :CHAR);
(*-----------------------------------------------------------------*)
  (*.............................................................*)
  PROCEDURE MERGE(ROW :INTEGER; S :STRING);
  (*.............................................................*)
  VAR COL: INTEGER;
  BEGIN
   FOR COL :=1 TO MAXCOL DO NEWARRAY[ROW+HEIGHT,COL+WIDTH] :=S[COL]='X';
  END; (* MERGE *)

BEGIN
 CASE SYMBOL OF
 '1': BEGIN
        MAXCOL :=4;
       MERGE(3,'  X ');
       MERGE(2,' X  ');
       MERGE(1,'X   ');
     END;
 '2': BEGIN
       MAXCOL :=4;
       MERGE(3,' X  ');
       MERGE(2,'  X ');
       MERGE(1,'   X ');
     END;
 '3': BEGIN
       MAXCOL :=4;
       MERGE(1,'XXXX');
     END;
 '+': BEGIN
        MAXCOL :=5;
       MERGE(5,'  X  ');
       MERGE(4,'  X  ');
       MERGE(3,'XXXXX');
       MERGE(2,'  X  ');
       MERGE(1,'  X  ');
     END;
 'H': BEGIN
       MAXCOL :=5;
       MERGE(5,'X   X');
       MERGE(4,'X   X');
       MERGE(3,'XXXXX');
       MERGE(2,'X   X');
       MERGE(1,'X   X');
     END;
 'O': BEGIN
       MAXCOL :=8;
       MERGE(8,'  XXXX  ');
       MERGE(7,' X    X ');
       MERGE(6,'X      X');
       MERGE(5,'X      X');
       MERGE(4,'X      X');
       MERGE(3,'X      X');
       MERGE(2,' X    X ');
       MERGE(1,'  XXXX  ');
     END;
  END; (*CASE*)
END; (* INIT *)
```

```
BEGIN (* DEFINEMOLEC *)
 OX :=(32-8) DIV 2 -1 ; OY :=(32-8) DIV 2;
 H1X :=0; H1Y :=0;
 H2X :=0; H2Y :=32-5;
 H3X :=32-5; H3Y :=(32-5) DIV 2;
 CASE ACIDITY OF
  BASIC: BEGIN
           INIT(OX,0,'O');
           INIT(H3X,1,'H');
           INIT(22,3,'3'); (* bond *)
           INIT(20,10,'3'); (* - sign *)
         END;
  NEUTRAL :CASE SHAPE OF
        1 : BEGIN
             INIT(OX,OY,'O');
             INIT(H3X,H3Y,'H');
             INIT(22,16,'3');
             INIT(H1X,H1Y,'H');
             INIT(6,6,'1');
            END;
        2 : BEGIN
            INIT(OX,0,'O');
            INIT(H3X,3,'H');
            INIT(22,6,'3'); (* bond *)
            INIT(0,H2Y-OY,'H');
            INIT(6,22-OY,'2'); (* add H + bond *)
           END;
        END;
  ACIDIC : BEGIN
            INIT(OX,OY,'O');
            INIT(H3X,H3Y,'H');
            INIT(22,16,'3');
            INIT(H1X,H1Y,'H');
            INIT(6,6,'1');
            INIT(H2X,H2Y,'H');
            INIT(6,22,'2');
            INIT(0,14,'+');
          END;
  END; (*CASE*)
END; (* DEFINEMOLEC  *)

(*==================================================================*)
PROCEDURE MOVEMOLECULE(ANYARRAY :BIGSIZE; HEIGHT :INTEGER;
                        VAR  X,Y,DX,DY :INTEGER);
(*==================================================================*)
BEGIN
 DRAWBLOCK(ANYARRAY,4,0,0,32,HEIGHT,X,Y,MODE);
 X :=X+DX;
 Y :=Y+DY;
 DRAWBLOCK(ANYARRAY,4,0,0,32,HEIGHT,X,Y,MODE);
END; (* MOVEMOLECULE *)


(*$I :2LITMUS*)
(*$I :3LITMUS*)
(*$I :4LITMUS*)
```

```
(* 2LITMUS*)
   (*=================================================================*)
   PROCEDURE FALSEARRAY(VAR NEWARRY:BIGSIZE);
   (*=================================================================*)
   CONST MAX=32;
   VAR  ROW,COL : INTEGER;
   BEGIN
     FOR ROW:=1 TO MAX DO  FOR COL:=1 TO MAX DO  NEWARRY[ROW,COL]:=FALSE;
   END; (* FALSEARRAY *)

   (*=================================================================*)
   PROCEDURE EXPLAIN;
   (*=================================================================*)
   VAR CH:CHAR;
   BEGIN
     INITTURTLE;
     WSTAT(50,160,'Litmus is a dye.');
     WSTAT(50,140,'It is composed of  a mixture of');
     WSTAT(10,125,'many complex organic molecules.');
     WSTAT(50,105,'To simplify reactions involving ');
     WSTAT(0,90,'litmus, a litmus molecule will be');
     WSTAT(0,75,'represented by the following');
     WSTAT(0,60,'structure:-');
     DRAWLITMUS(100,0,20,BLUECOL);
     DRAWEL(100,0,20,BLUECOL);
     GETKEY(CH,[SPACE,'Q']);
     IF NOT QUIT THEN DISPLAYLITMUS;
     IF NOT QUIT THEN CHANGELIT(80,80,24);
   END; (* EXPLAIN *)

   (*=================================================================*)
   PROCEDURE REACTION;
   (*=================================================================*)
   CONST  HPOSY=27; (*32-5*)
   VAR   LASTY,MINLEVEL,ACIDLEVEL,LITX, TOPY, SIZE,
         NUM, INCREASE,  X,Y,DX, DY : INTEGER;

     (*---------------------------------------------------------------*)
     PROCEDURE REACTINACID(ANYX,ANYY,SSIZE:INTEGER;  ACOL:SCREENCOLOR);
     (*---------------------------------------------------------------*)
     VAR K,ANYNUM:INTEGER; CH:CHAR;
       (*..............................................................*)
       PROCEDURE AREACTION(X1,Y1,SIZE:INTEGER);
       (*..............................................................*)
       VAR  CH:CHAR;
            J,NUMM,RTX,RTY,HX,HY : INTEGER;

         PROCEDURE ACIDNEUTRAL(VAR X,Y,DX,DY : INTEGER);
         BEGIN
          DRAWBOND(RTX+1,RTY,HX-3,RTY,WHITE2);
          DRAWBLOCK(LARGACID,4,0,0,32,32,X,Y,MODE); (* erase acid *)
          DRAWBLOCK(SMALLH,2,0,0,8,5,X,Y+HPOSY,MODE);
          DX:=-DX;  DY:=-DY;
          X:=X+DX;  Y:=Y+DY;
          DRAWBLOCK(WATER,4,0,0,32,24,X,Y,MODE);  (*display water *)
         END; (* ACIDNEUTRAL *)
```

```
      PROCEDURE INITPOS;
      BEGIN
         RTX:=X1+(3*SIZE);
         HX:=RTX + SIZE;
         RTY:=Y1 + SIZE;
         HY:=RTY - 3;  (* 3 is approx half ht of small H atom *)
         DX:=-8;  DY:=0;
         X:=HX - (NUMM*DX);
         Y:=HY - (NUMM*DY)-HPOSY;
      END;  (* INITPOS *)

   BEGIN  (* AREACTION *)
     NUMM:=4;
     INITPOS;
     INITSMH(SMALLH);
     DRAWBLOCK(LARGACID,4,0,0,32,32,X,Y,MODE); (*display acid*)
     FOR J:=1 TO NUMM DO
      BEGIN
         MOVEMOLECULE(LARGACID,32,X,Y,DX,DY);
         WAIT(50);
      END;
     ACIDNEUTRAL(X,Y,DX,DY);      (* replace acid with water *)

     DRAWLITMUS(X1,Y1,SIZE,REDCOL);
     FOR J:=1 TO NUMM DO
      BEGIN
        MOVEMOLECULE(WATER,24,X,Y,DX,DY);
        WAIT(50);
      END;
     DRAWBLOCK(WATER,4,0,0,32,24,X,Y,MODE); (*erase water*)
   END;  (* AREACTION *)

BEGIN  (* REACTINACID *)
  IF ACOL=BLUECOL  THEN
  BEGIN
     ANYNUM:=3;
     ANYY:=ANYY+(ANYNUM-1)*INCREASE;
     FOR K:=1 TO ANYNUM DO
      BEGIN
        AREACTION(ANYX,ANYY,SSIZE);
        ANYY:=ANYY-INCREASE;
        WAIT(300);
      END;
   END;
END;  (*REACTINACID*)

(*-----------------------------------------------------------------------*)
PROCEDURE REACTINBASE(ANYX,ANYY,SSIZE:INTEGER;  ACOL:SCREENCOLOR);
(*-----------------------------------------------------------------------*)
VAR   K,ANYNUM:INTEGER;

  (*.....................................................................*)
  PROCEDURE BREACTION(X1,Y1,SIZE:INTEGER);
  (*.....................................................................*)
  VAR   CH:CHAR;  J,NUMM,RTX,RTY,HX,HY:INTEGER;
    PROCEDURE BASENEUTRAL(VAR X,Y,DX,DY  :  INTEGER);
```

```
BEGIN
 DRAWBLOCK(HYDROX,4,0,0,32,16,X,Y,MODE); (* erase base *)
 ADDH(X1,Y1,SIZE,BLACK1);  (* erase H on litmus *)
 DRAWBLOCK(WATER,4,0,0,32,24,X,Y,MODE); (*display water at new pos*)
 DX:=-DX; DY:=-DY;
END; (* BASENEUTRAL *)

PROCEDURE INITPOS;
BEGIN
   RTX:=X1+(3*SIZE);
   HX:=RTX + SIZE ;
   RTY:=Y1 + SIZE;
   HY:=RTY - 3; (* 3 is approx half ht of H atom*)
    DX:=-8; DY:=0;
    X:=HX - (NUMM*DX);
    Y:=HY - (NUMM*DY)-17;
  END; (* INITPOS *)

BEGIN (* BREACTION *)
 NUMM:=4;
 INITPOS;
 DRAWBLOCK(HYDROX,4,0,0,32,32,X,Y,MODE);
 FOR J:=1 TO NUMM DO
  BEGIN
    MOVEMOLECULE(HYDROX,16,X,Y,DX,DY); WAIT(50);
  END;
 BASENEUTRAL(X,Y,DX,DY);
 DRAWLITMUS(X1,Y1,SIZE,BLUECOL);(* CHANGE COLOUR *)
 FOR J:=1 TO NUMM DO
  BEGIN
   MOVEMOLECULE(WATER,24,X,Y,DX,DY); WAIT(50);
  END;
 DRAWBLOCK(WATER,4,0,0,32,32,X,Y,MODE);
END; (* BREACTION *)


BEGIN  (* REACTINBASE *)
 IF ACOL=REDCOL THEN
  BEGIN
   ANYNUM:=3;
   ANYY:=ANYY+ (ANYNUM-1)*INCREASE;
   FOR K:=1 TO ANYNUM DO
    BEGIN
     BREACTION(ANYX,ANYY,SSIZE);
     ANYY:=ANYY-INCREASE;
     WAIT(500);
    END;
  END;
END;(*REACTINBASE*)

(*---------------------------------------------------------------------*)
PROCEDURE SHOWLIT(X,Y,SIZE,INCREASE,ANYNUM:INTEGER;
            VAR LASTY:INTEGER; COL:SCREENCOLOR);
(*---------------------------------------------------------------------*)
VAR J  : INTEGER;
```

```
BEGIN
 FOR J:=1 TO ANYNUM DO
  BEGIN
   DRAWLITMUS(X,Y,SIZE,COL);
   IF COL=REDCOL THEN ADDH(X,Y,SIZE,WHITE1);
   Y:=Y-INCREASE;
  END; (* FOR *)
 LASTY:=Y+INCREASE;
END; (* SHOWLIT *)

(*-----------------------------------------------------------------------*)
PROCEDURE MOVELIT(X,TOPY,INCREASE:INTEGER;   VAR LASTY:INTEGER;
                                                COL:SCREENCOLOR);
(*-----------------------------------------------------------------------*)
VAR J: INTEGER;  SURFACE:BOOLEAN;

  (*.....................................................................*)
  PROCEDURE BREAKSURFACE(X,SIZE:INTEGER);
  (*.....................................................................*)
  BEGIN
    SURFACE:=FALSE;
    DRAWLINE(X-SIZE,ACIDLEVEL,X+3*SIZE,ACIDLEVEL,BLACK2);
  END;

BEGIN
 SURFACE:=TRUE;
 REPEAT
  LASTY:=LASTY-INCREASE;
  IF LASTY<=ACIDLEVEL THEN
      BEGIN  IF SURFACE THEN BREAKSURFACE(X,SIZE); END;
  DRAWLITMUS(X,LASTY,SIZE,COL);
  IF COL=REDCOL THEN    (*draw litshape molecule beneath*)
    BEGIN               (*display H on bottom molecule*)
     ADDH(X,LASTY,SIZE,WHITE1);
     ADDH(X,TOPY,SIZE,BLACK1);
    END;                (*erase H from top molecule*)
  DRAWLITMUS(X,TOPY,SIZE,BLACK2); (* erase top molecule*)
  TOPY:=TOPY-INCREASE;
 UNTIL LASTY<MINLEVEL;
END; (* MOVELIT *)

(*-----------------------------------------------------------------------*)
PROCEDURE DRAWLEVEL(LEVEL:INTEGER);
(*-----------------------------------------------------------------------*)
BEGIN
 DRAWLINE(10,LEVEL,260,LEVEL,WHITE2);
END;

(*-----------------------------------------------------------------------*)
PROCEDURE TITLE(ANYCOL:SCREENCOLOR; ANYTYPE:PH);
(*-----------------------------------------------------------------------*)
VAR  S1,S2: STRING;  CH:CHAR;
BEGIN
 IF ANYCOL=REDCOL THEN S1:='RED' ELSE S1:='BLUE';
 CASE ANYTYPE OF
```

```
    ACIDIC:  S2:='an acidic';
    BASIC:   S2:='a basic';
   END; (*CASE*)
  S1:=CONCAT('Press <SPACE BAR> to place ',S1,' litmus');
  S2:=CONCAT('in ',S2,' solution');
  WSTAT(1,10,S1);
  WSTAT(1,0,S2);
  GETKEY(CH,[SPACE,'Q']);
  WSTAT(1,10,S1);
  WSTAT(1,0,S2);   (* erase *)
END;

(*-----------------------------------------------------------------*)
PROCEDURE RESULT(ANYCOL:SCREENCOLOR; ANYTYPE:PH);
(*-----------------------------------------------------------------*)
VAR S : STRING;   CH:CHAR;

  (*...............................................................*)
  PROCEDURE PROMPTSPACE;
  (*...............................................................*)
  BEGIN
    WSTAT(20,181,'Press <SPACE BAR> to continue');
  END;

BEGIN
 FILLBOX(XMIN,XMAX,YMIN,YMIN+10,BLACK1);
 CASE ANYTYPE OF
  ACIDIC: IF ANYCOL=REDCOL THEN   S:='RED litmus unchanged in acidic solution'
          ELSE    S:='BLUE litmus turns RED in acidic solution';
  BASIC: IF ANYCOL=REDCOL THEN S:='RED litmus turns BLUE in basic solution'
          ELSE  S:='BLUE litmus unchanged in basic solution';
  END;(*CASE*)
 WSTAT(1,2,S);
 PROMPTSPACE;
 GETKEY(CH,[SPACE,'Q']);
 PROMPTSPACE;
END; (* RESULT *)

(*-----------------------------------------------------------------*)
PROCEDURE SHOWSOLN(ACIDITY:PH; MAXHT:INTEGER);
(*-----------------------------------------------------------------*)
VAR  J: INTEGER;  X,Y :ARRAY[1..4] OF INTEGER;
BEGIN
   X[1]:=1; Y[1]:=25; X[2]:=70; Y[2]:=MAXHT-36;
   X[3]:=XMAX-70; Y[3]:=19; X[4]:=XMAX-33; Y[4]:=MAXHT-33;
   IF ACIDITY=ACIDIC THEN
    FOR J:=1 TO 4 DO DRAWBLOCK(LARGACID,4,0,0,32,32,X[J],Y[J],MODE)
      ELSE
       FOR J:=1 TO 4 DO DRAWBLOCK(HYDROX,4,0,0,32,16,X[J],Y[J],MODE);
END; (* SHOWSOLN *)

(*-----------------------------------------------------------------*)
PROCEDURE DISPLAY(ACOL:SCREENCOLOR; SOLN:PH);
(*-----------------------------------------------------------------*)
BEGIN
 INITTURTLE;
```

```
        DRAWLEVEL(ACIDLEVEL);
        SHOWLIT(LITX,TOPY,SIZE,INCREASE,NUM,LASTY,ACOL);
        SHOWSOLN(SOLN,ACIDLEVEL);
        TITLE(ACOL,SOLN);
        IF QUIT THEN EXIT(DISPLAY);
        MOVELIT(LITX,TOPY,INCREASE,LASTY,ACOL);
        IF SOLN=ACIDIC THEN  REACTINACID(LITX,LASTY,SIZE,ACOL)
          ELSE  REACTINBASE(LITX,LASTY,SIZE,ACOL);
        RESULT(ACOL,SOLN);
      END; (* DISPLAY *)

    BEGIN  (* REACTION *)
      INITSMH(SMALLH);
      LITX:=120; TOPY:=175; SIZE:=6; NUM:=6;
      INCREASE:=2*SIZE+1;
      ACIDLEVEL:=TOPY-((NUM*2+1)*SIZE)+ SIZE DIV 2;
      MINLEVEL:=ACIDLEVEL-6*SIZE;
      WATER:=BLANK;
      DEFINEMOLEC(WATER,NEUTRAL,1); (*water shape for acidic solns*)
      DISPLAY(BLUECOL,ACIDIC);
      IF NOT QUIT THEN DISPLAY(REDCOL,ACIDIC);
      WATER:=BLANK;
      DEFINEMOLEC(WATER,NEUTRAL,2); (*water shape for basic solutions *)
      IF NOT QUIT THEN DISPLAY(BLUECOL,BASIC);
      IF NOT QUIT THEN DISPLAY(REDCOL,BASIC);
    END; (* REACTION *)

    (*==============================================================*)
    PROCEDURE INITVARS;
    (*==============================================================*)
    BEGIN
      BLUECOL:=BLUE;
      IF COLOUR THEN REDCOL:=VIOLET  ELSE REDCOL:=WHITE2;
      FALSEARRAY(BLANK);
      LARGACID:=BLANK;
      DEFINEMOLEC(LARGACID,ACIDIC,0);
      HYDROX:=BLANK;
      DEFINEMOLEC(HYDROX,BASIC,0);
    END; (* INITVARS *)


BEGIN  (* MICRO *)
  INITVARS;
  IF NOT  QUIT THEN
   BEGIN
     EXPLAIN;
     IF NOT QUIT THEN REACTION;
     TEXTMODE;
   END ;
  PAGE(OUTPUT);
END; (*MICRO*)
```

```
(* 3LITMUS *)
(***********************************************************)
PROCEDURE MACRO;
(***********************************************************)
TYPE
  ACIDITY=(ACIDIC,BASIC);
  LITCOL=(REDLIT,BLUELIT);
  LITTYPE=RECORD
            X,Y : INTEGER;
            COL : SCREENCOLOR;
          END;
VAR  REDCOL,BLUECOL : SCREENCOLOR;

   (*============================================================*)
   PROCEDURE DRAWBOX(X1,Y1,X2,Y2:INTEGER; COL:SCREENCOLOR);
                    (*from bottom L.H.corner to top R.H.corner*)
   (*============================================================*)
   BEGIN
    MOVECOL(X1,Y1,COL);
    MOVETO(X2,Y1);
    MOVETO(X2,Y2);
    MOVETO(X1,Y2);
    MOVECOL(X1,Y1,NONE);
   END; (* DRAWBOX *)

   (*============================================================*)
   PROCEDURE DRAWBEAKER(X,Y,SIZE:INTEGER; COL:SCREENCOLOR);
   (*============================================================*)
   (* x,y is coord. for the bottom L.H.corner of beaker*)
   VAR EDGE  : INTEGER;
   BEGIN
     EDGE:=SIZE DIV 12;
     MOVECOL(X-EDGE,Y+SIZE+EDGE,COL);
     MOVETO(X,Y+SIZE);
     MOVETO(X,Y);
     MOVETO(X+SIZE,Y);
     MOVETO(X+SIZE,Y+SIZE);
     MOVECOL(X+SIZE+EDGE,Y+SIZE+EDGE,NONE);
   END; (* DRAWBEAKER *)

   (*============================================================*)
   PROCEDURE DRAWTTUBE(X,Y,WIDTH,LENGTH,LEVEL:INTEGER;   COL:SCREENCOLOR);
   (*============================================================*)
   (* x,y is bottom L.H. corner of test tubelevel of soln in ttube. To get good curve on
      bottom of tube make the width a multiple  of 8 *)
   VAR EIGHTH,SIXNTH, RWIDTH : REAL;
       CURVEX,CURVEY: ARRAY[1..15] OF INTEGER; (*bottom of tube*)
       J : INTEGER;
   BEGIN
    RWIDTH:=WIDTH;   (* convert integer width to a real*)
    EIGHTH:=RWIDTH/8; (* eighth the width of TTUBE*)
    SIXNTH:=RWIDTH/16; (*sixteenth the width of TTUBE*)

    CURVEY[1]:=ROUND(EIGHTH*0.5);  (* necessary for curve at *)
    CURVEY[2]:=ROUND(EIGHTH*0.75); (* bottom of ttube *)
    CURVEY[3]:=ROUND(EIGHTH);
```

```
 CURVEY[4]:=ROUND(EIGHTH*1.25);
 CURVEY[5]:=ROUND(EIGHTH*1.32);
 CURVEY[6]:=ROUND(EIGHTH*1.40);
 CURVEY[7]:=ROUND(EIGHTH*1.50);
 CURVEY[8]:=CURVEY[7];
 FOR J:=1 TO 7 DO  CURVEY[16-J]:=CURVEY[J];
 FOR J:=1 TO 15 DO CURVEX[J]:=ROUND(SIXNTH*J);
 MOVECOL(X,Y+LENGTH,COL);
 MOVETO(X,Y);
 FOR J:=1 TO 15 DO MOVETO(X+CURVEX[J],Y-CURVEY[J]);
 MOVETO(X+WIDTH,Y);
 MOVETO(X+WIDTH,Y+LENGTH);
 MOVETO(X+WIDTH,Y + LEVEL);  (* draw liq. level *)
 MOVECOL(X,Y+LEVEL,NONE);
END; (* DRAWTTUBE *)

(*==============================================================*)
PROCEDURE NAME(X,Y:INTEGER; ANYSOLN:ACIDITY);
(*==============================================================*)
VAR S:STRING;
BEGIN
   IF ANYSOLN=ACIDIC THEN S:='ACID' ELSE S:='BASE';
   WSTAT(X,Y,S);
END;

(*==============================================================*)
PROCEDURE FILLBEAKER(X,Y,WIDTH,LEVEL:INTEGER;COL:SCREENCOLOR);
(*==============================================================*)
BEGIN
  REPEAT
    Y:=Y+2;
    DRAWLINE(X+3,Y,X+WIDTH-2,Y,COL);     (*display*)
    WAIT(40);
    DRAWLINE(X+3,Y,X+WIDTH-2,Y,BLACK2);  (* erase *)
  UNTIL Y>=LEVEL;
 DRAWLINE(X+2,Y,X+WIDTH-2,Y,COL);
END; (* FILLBEAKER *)

(*==============================================================*)
PROCEDURE LITMUSPAPER;
(*==============================================================*)
VAR BEAKERY,BEAKERX,BSIZE, LITWIDTH,LITHT,LEVEL :INTEGER;
      SOLN: ACIDITY;
   (*-----------------------------------------------------------*)
   PROCEDURE INITCONST;
   (*-----------------------------------------------------------*)
   VAR CENTRX:INTEGER;
   BEGIN
     CENTRX:=140;
     BEAKERY:=25;
     BSIZE:=90;
     BEAKERX:=CENTRX-(BSIZE DIV 2);
     LITWIDTH:=10;
     LITHT:=40;
     LEVEL:=BEAKERY+(2*BSIZE DIV 3);
   END; (* INITCONST *)
```

```
(*---------------------------------------------------------------*)
PROCEDURE TESTLITMUS(ASOLN:ACIDITY);
(*---------------------------------------------------------------*)
VAR LITMUS:ARRAY[REDLIT..BLUELIT] OF LITTYPE;
    LIT:LITCOL;
    CHANGE: BOOLEAN;
    CH: CHAR;
  (*............................................................*)
  PROCEDURE INITVARS;
  (*............................................................*)
  VAR SPACING : INTEGER;  ANYLIT : LITCOL;
  BEGIN
    SPACING:=(BSIZE-(2*LITWIDTH)) DIV 3;
    LITMUS[REDLIT].X:=BEAKERX  + SPACING;
    LITMUS[BLUELIT].X:=LITMUS[REDLIT].X + LITWIDTH +SPACING;
    LITMUS[REDLIT].COL:=REDCOL;
    LITMUS[BLUELIT].COL:=BLUECOL;
    FOR ANYLIT:=REDLIT TO BLUELIT DO
       LITMUS[ANYLIT].Y:=BEAKERY + BSIZE + 25;
  END; (* INITVARS *)

  (*............................................................*)
  PROCEDURE DRAWLITMUS(X,Y:INTEGER; COL:SCREENCOLOR);
  (*............................................................*)
  BEGIN
   VIEWPORT(X,X+LITWIDTH,Y,Y+LITHT);
   FILLSCREEN(COL);
   VIEWPORT(XMIN,XMAX,YMIN,YMAX);
  END; (* DRAWLITMUS *)

  (*............................................................*)
  PROCEDURE SHOWLITMUS;
  (*............................................................*)
  VAR LIT : LITCOL;
  BEGIN
   FOR LIT:=REDLIT TO BLUELIT DO
    WITH LITMUS[LIT] DO
     BEGIN
      DRAWLITMUS(X,Y,COL);
      IF NOT COLOUR THEN
       BEGIN
        IF LIT=REDLIT THEN
         WSTAT(X-25,Y+15,'Red')
         ELSE
         WSTAT(X+25,Y+15,'Blue');
       END;
     END;
  END;  (* SHOWLITMUS *)
  (*............................................................*)
  PROCEDURE MOVELIT(ANYLIT:LITCOL);
  (*............................................................*)
  CONST  DROP=10;
  VAR NEWY: INTEGER;
```

```
(*-------------------------------------------*)
PROCEDURE ERASE(X1,Y1,X2,Y2:INTEGER);
(*-------------------------------------------*)
BEGIN
 FILLBOX((X1,X2,Y1,Y2,BLACK2);
END; (* ERASE *)

BEGIN
 WITH LITMUS[ANYLIT] DO
 REPEAT
  NEWY:=Y-DROP;
  DRAWLITMUS(X,NEWY,COL);
  ERASE(X,NEWY+LITHT,X+LITWIDTH,Y+LITHT);
  Y:=NEWY;
 UNTIL(Y+10)<LEVEL;
END; (* MOVELIT *)

(*.............................................*)
PROCEDURE CHANGECOL(ANYLIT:LITCOL);
(*.............................................*)
BEGIN
 WITH LITMUS[ANYLIT] DO
  BEGIN
   IF COL=REDCOL THEN COL:=BLUECOL  ELSE COL:=REDCOL;
   FILLBOX(X,X+LITWIDTH,Y,LEVEL,COL);
  END;
END; (* CHANGECOL *)

(*.............................................*)
PROCEDURE CHECKLIT(ANYLIT:LITCOL);
(*.............................................*)
BEGIN
 CHANGE:=((ANYLIT=REDLIT) AND (ASOLN=BASIC));
 IF NOT CHANGE THEN CHANGE:=((ANYLIT=BLUELIT) AND (ASOLN=ACIDIC));
END; (* CHECKLIT *)

(*.............................................*)
PROCEDURE STATEMENT(ANYSOLN:ACIDITY);
(*.............................................*)
VAR S:STRING;
BEGIN
  FILLBOX(XMIN,XMAX,YMIN,YMIN+10,WHITE1);
  IF ASOLN=BASIC THEN S:='BASE TURNS LITMUS BLUE'
     ELSE  S:='ACID TURNS LITMUS RED';
  CHARTYPE(5);
  WSTAT(60,0,S);
  CHARTYPE(MODE);
END; (* STATEMENT *)

(*.............................................*)
PROCEDURE PROMPT(ANYSOLN:ACIDITY);
(*.............................................*)
VAR S1,S2:STRING; CH:CHAR;
BEGIN
 FILLBOX(XMIN,XMAX,YMIN,YMIN+10,BLACK1);
 IF ANYSOLN=BASIC THEN S2:='a BASIC'  ELSE S2:='an ACIDIC';
```

```
        S1:='Press <SPACE BAR> to test';
        S2:=CONCAT(S2,' solution with litmus.');
        WSTAT(20,10,S1);
        WSTAT(2,0,S2);
        GETKEY(CH,[SPACE,'Q']);
        WSTAT(20,10,S1);
        WSTAT(2,0,S2);
       END;  (* PROMPT *)

    BEGIN  (* TESTLITMUS *)
      INITVARS;
      INITTURTLE;
      DRAWBEAKER(BEAKERX,BEAKERY,BSIZE,WHITE2);
      FILLBEAKER(BEAKERX,BEAKERY,BSIZE,LEVEL,WHITE2);
      NAME(BEAKERX+BSIZE+10,BEAKERY+(BSIZE DIV 2),ASOLN);
                              (*DISPLAY NAME*)
      SHOWLITMUS;
      PROMPT(ASOLN);
      IF QUIT THEN EXIT(LITMUSPAPER);
      FOR LIT:=REDLIT TO BLUELIT DO
       BEGIN
         MOVELIT(LIT);
         CHECKLIT(LIT);
         IF CHANGE THEN CHANGECOL(LIT);
       END;
      STATEMENT(ASOLN);
      GETKEY(CH,[SPACE,'Q']);
      IF QUIT THEN EXIT(LITMUSPAPER);
      NAME(BEAKERX+BSIZE+10,BEAKERY+(BSIZE DIV 2),ASOLN);  (*erase name*)
    END; (* TESTLITMUS *)


  BEGIN  (* LITMUSPAPER *)
   INITCONST;
   FOR SOLN:=ACIDIC TO BASIC DO TESTLITMUS(SOLN);
  END; (* LITMUSPAPER *)

(* 4LITMUS.TEXT *)
  (*==============================================================*)
  PROCEDURE TESTSOLN;
  (*==============================================================*)
  TYPE  SMSIZE=PACKED ARRAY[1..8,1..8] OF BOOLEAN;
  VAR  BSIZE,LTX,LTY,RTX,RTY: INTEGER; (* coord. of 2 beakers*)
     DROP: SMSIZE;
     LEVEL :INTEGER;
     SOLN: ACIDITY;
     LIT:LITCOL;
     LITMUS:ARRAY[REDLIT..BLUELIT] OF LITTYPE;
     CH:CHAR;

    (*------------------------------------------------------------*)
    PROCEDURE INITCONST;
    (*------------------------------------------------------------*)
      (*.........................................................*)
      PROCEDURE INITDROP;
      (*.........................................................*)
```

```
      PROCEDURE SMALLBITS(ROW : INTEGER; VAR BITS :SMSIZE;S :STRING);
      VAR COL : INTEGER;
      BEGIN
         FOR COL :=1 TO 8 DO BITS[ROW ,COL] :=S[COL]='X';
      END; (* SMALLBITS *)

   BEGIN
     SMALLBITS(8,DROP,'   X    ');
     SMALLBITS(7,DROP,'  XXX   ');
     SMALLBITS(6,DROP,' XXXXX  ');
     SMALLBITS(5,DROP,'XXXXXXX ');
     SMALLBITS(4,DROP,'XXXXXXXX');
     SMALLBITS(3,DROP,'XXXXXXXX');
     SMALLBITS(2,DROP,' XXXXXX ');
     SMALLBITS(1,DROP,'  XXXX  ');
   END; (* INITDROP *)

BEGIN
  BSIZE :=70;
  LTY :=50; RTY :=50;
  LTX :=(XMAX DIV 4) - (BSIZE DIV 2);
  RTX :=(3*XMAX DIV 4)- (BSIZE DIV 2);
  LEVEL :=LTY+(2*BSIZE DIV 3);
  INITDROP;
END; (* INITCONST *)

(*----------------------------------------------------------------*)
PROCEDURE INITVARS;
(*----------------------------------------------------------------*)
BEGIN
    LITMUS[REDLIT].X :=LTX+(BSIZE DIV 2);
    LITMUS[BLUELIT].X :=RTX+(BSIZE DIV 2);
    LITMUS[REDLIT].Y :=YMAX-10;
    LITMUS[BLUELIT].Y :=LITMUS[REDLIT].Y;
    LITMUS[REDLIT].COL :=REDCOL;
    LITMUS[BLUELIT].COL :=BLUECOL;
 END; (* INITVARS *)

(*----------------------------------------------------------------*)
PROCEDURE PROMPT(ANYLIT :LITCOL);
(*----------------------------------------------------------------*)
VAR S1,S2 :STRING; CH :CHAR;
BEGIN
  FILLBOX(XMIN,XMAX,YMIN,YMIN+10,BLACK1);
  IF ANYLIT=BLUELIT THEN S2 :='a BASIC' ELSE S2 :='an ACIDIC';
  S1 :='Press <SPACE BAR> to test';
  S2 :=CONCAT(S2,' solution with litmus.');
  WSTAT(0,10,S1);
  WSTAT(0,0,S2);
  GETKEY(CH,[SPACE,'Q']);
  WSTAT(0,10,S1);
  WSTAT(0,0,S2);
END; (* PROMPT *)
```

```
(*---------------------------------------------------------------*)
PROCEDURE DRAWDROP(ANYLIT :LITCOL);
(*---------------------------------------------------------------*)
BEGIN
 WITH LITMUS[ANYLIT] DO DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);
END;

(*---------------------------------------------------------------*)
PROCEDURE MOVEDROP(ANYLIT :LITCOL);
(*---------------------------------------------------------------*)
BEGIN
 WITH LITMUS[ANYLIT] DO
  BEGIN
   REPEAT
    DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);Y:=Y-6;  (*erase *)
    DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);         (*display *)
    WAIT(30);
   UNTIL (Y<LEVEL);
   DRAWBLOCK(DROP,2,0,0,8,8,X,Y,MODE);          (*erase in soln*)
  END; (*WITH*)
END; (* MOVEDROP *)

(*---------------------------------------------------------------*)
PROCEDURE CHANGECOL(ANYLIT :LITCOL);
(*---------------------------------------------------------------*)
VAR  X1,X2,Y : INTEGER;
     COL : SCREENCOLOR;
BEGIN
 IF ANYLIT=REDLIT THEN
  BEGIN
   X1:=LTX+2; X2:=X1+BSIZE-3; Y:=LTY+1; COL:=REDCOL;
  END
  ELSE
  BEGIN
   X1:=RTX+2; X2:=X1+BSIZE-3; Y:=RTY+1; COL:=BLUECOL;
  END;
  FILLBOX(X1,X2,Y,LEVEL,COL);
END; (* CHANGECOL *)

(*---------------------------------------------------------------*)
PROCEDURE STATEMENT;
(*---------------------------------------------------------------*)
BEGIN
 FILLBOX(XMIN,XMAX,YMIN,YMIN+20,WHITE1);
 CHARTYPE(5);
 WSTAT(50,10,'ACID TURNS LITMUS RED');
 WSTAT(50,0,'BASE TURNS LITMUS BLUE');
 CHARTYPE(MODE);
END; (* STATEMENT *)

BEGIN (* TESTSOLN *)
 INITCONST;
 INITVARS;
 INITTURTLE;
 DRAWBEAKER(LTX,LTY,BSIZE,WHITE1);
 FILLBEAKER(LTX,LTY,BSIZE,LEVEL,WHITE1);
```

```
    NAME(LTX+(BSIZE DIV 2)-16,LTY-12,ACIDIC);
    DRAWBEAKER(RTX,RTY,BSIZE,WHITE2);
    FILLBEAKER(RTX,RTY,BSIZE,LEVEL,WHITE2);
    NAME(RTX+(BSIZE DIV 2)-16,RTY-12,BASIC);
    FOR LIT:=REDLIT TO BLUELIT DO
     BEGIN
       DRAWDROP(LIT);
       PROMPT(LIT);
       IF QUIT THEN EXIT(TESTSOLN);
       MOVEDROP(LIT);
       CHANGECOL(LIT);
     END;
    STATEMENT;
    GETKEY(CH,[SPACE,'Q']);
  END;  (* TESTSOLN *)

BEGIN  (* MACRO *)
  BLUECOL:=BLUE;
  IF COLOUR THEN REDCOL:=VIOLET ELSE REDCOL:=WHITE2;
  IF NOT QUIT THEN LITMUSPAPER;
  IF NOT QUIT THEN TESTSOLN;
END; (*MACRO*)

(*****************************************************************)
PROCEDURE INTRO;
(*****************************************************************)
VAR X,Y: INTEGER; CH:CHAR;
BEGIN
 PAGE(OUTPUT);
 X:=0; Y:=4;
 WRITE(AT(X,Y),'LITMUS is an acid/base indicator which'); Y:=Y+2;
 WRITE(AT(X,Y),'is usually used in the form of LITMUS');   Y:=Y+2;
 WRITE(AT(X,Y),'PAPER  ie. paper which is impregnated');  Y:=Y+2;
 WRITE(AT(X,Y),'with litmus.');  Y:=Y+4;
 WRITE(AT(X,Y),'It is used as a simple test to determine');  Y:=Y+2;
 WRITE(AT(X,Y),'if a solution is acidic or basic.');
 GOTOXY(40,Y);
 GETKEY(CH,[SPACE,'Q']);
 PAGE(OUTPUT);
END; (* INTRO  *)

(*****************************************************************)
PROCEDURE SELECT(VAR CH:CHAR);
(*****************************************************************)
CONST DEMO='SCOPIC demonstration     ......(';
VAR  X,Y: INTEGER;
BEGIN
 TEXTMODE;
 PAGE(OUTPUT);
 X:=0; Y:=1;
 WRITE(AT(X,Y),AROW(40,'*')); Y:=Y+2;
 WRITE(AT(10,Y),'REACTIONS OF LITMUS'); Y:=Y+2;
 WRITE(AT(X,Y),AROW(40,'*'));  Y:=Y+4;;
 WRITE(AT(X,Y),CONCAT('MACRO',DEMO,'1)'));  Y:=Y+3;
 WRITE(AT(X,Y),CONCAT('MICRO',DEMO,'2)')); Y:=Y+3;
 WRITE(AT(X,Y),'QUIT - back to main menu      ......(Q)');  Y:=Y+3;
```

```
 WRITE(AT(X+10,Y),'Select option ..........(  )');
 GETTEXTCHAR(37,Y,CH,['1','2','Q']);
 QUIT:=CH='Q';
 PAGE(OUTPUT);
END; (* SELECT *)

(*********************************************************)
PROCEDURE CHECKCOL(VAR ACOLOUR:BOOLEAN);
(*********************************************************)
VAR   MONITOR: STRING;
BEGIN
 GETCVAL(MONITOR);
 ACOLOUR:=(MONITOR='INCOL');
END;


BEGIN  (* MAIN *)
 SETCHAIN(' :DEMOMENU');
 CHARTYPE(MODE);
 CHECKCOL(COLOUR);
 (*INTRO;*)
 REPEAT
  SELECT(OPTION);
  IF NOT QUIT THEN
   BEGIN
   CASE OPTION OF
    '1' : MACRO;
    '2' : MICRO;
    END; (*CASE*)
    QUIT:=FALSE;
   END;
 UNTIL QUIT;
END. (*LITMUS*)
```